



PLC Checker User Guide

Itris Automation Square

January 2011

<http://www.automationsquare.com>

VAS

Contents

1	Preamble	5
2	Introduction	6
3	Eclipse platform tutorial	7
3.1	Installation	7
3.1.1	Hardware and software configuration needed	7
3.1.2	Usage	7
3.1.3	Platform installation	8
3.2	Software update	10
3.3	Launching the platform	15
3.3.1	Workspace	15
3.3.2	Identification	15
3.3.3	Work bench	16
3.4	Import/Export files into/from the project	18
3.4.1	Drag'N Drop	18
3.4.2	Importing files	18
3.5	Comparing files	21
3.6	Local history	22
3.6.1	Compare with a older version	22
3.6.2	Replacing by an older version	23
4	PLC Checker project creation and usage	24
4.1	PLC Checker project creation	25
4.1.1	Default PLC Checker project creation with a wizard on Itris Automation Square website	25
4.1.2	Creation on IAS with wizard (PLC Checker project to be defined).	29
4.1.3	Creation on IAS without wizard.	32
4.1.4	New PLC Checker project creation on Eclipse platforme	36
4.1.5	Existing PLC Checker project creation on Eclipse platforme	38
4.2	Run PLC Checker	39
4.2.1	Run PLC Checker from an Eclipse platform :	39
4.2.2	Run PLC checker from the website:	42
4.3	View PLC Checker results	44
4.3.1	View results on Eclipse platform	44
5	PLC export files generation	46
5.1	FEF export file generation from PL7-Pro	46
5.2	L5K export file generation from RSLogix5000	46
5.3	ASC, AWL, GR7 and SCL Export files generation from Step7	47
5.3.1	Create ASC file	47
5.3.2	Create AWL file	49
5.3.3	Create GR7 file	53
5.3.4	Export files	55
5.4	XEF export file generation from Unity	56

6	Concepts	57
6.1	Rules	57
6.1.1	Generality	57
6.1.2	Rules activation	57
6.1.3	Rules name	57
6.1.4	Rules Identity	58
6.1.5	Code filter	58
6.1.6	Data filter	58
6.1.7	To enable a rule using other rules	58
6.1.8	To enable one rule or rules group in a list	59
6.1.9	Identifier for .csv format export	59
6.1.10	Node visibility for protected files	59
6.1.11	List of editable properties for protected files	59
6.1.12	Errors severity	59
6.1.13	"warningThreshold" property	59
6.1.14	"errorThreshold" property	59
6.1.15	"defaultResultId" property	59
6.1.16	"errorMessage" property	60
6.1.17	"noMatchMessage" property	60
6.2	Filters	60
6.2.1	Generality	60
6.2.2	Limitations	60
6.2.3	Symbols filters	60
6.2.4	Symbols filters using regular expressions	61
6.2.5	Attributes filters	61
6.2.6	Absolutes addresses filters	61
6.2.7	Filters intersection	62
6.2.8	Filters union	62
6.2.9	Filters difference	62
6.2.10	Filters usage	62
6.3	Result file	63
6.3.1	Printing results file	63
7	The rules	64
7.1	Rules group (rulesGroup)	64
7.1.1	Usage	64
7.1.2	Properties	64
7.2	Read Check (readCheck)	65
7.2.1	Usage	65
7.2.2	Properties	65
7.3	Write Check (writeCheck)	65
7.3.1	Usage	65
7.3.2	Properties	66
7.4	Symbol check (symbolCheck)	67
7.4.1	Usage	67
7.4.2	Properties	67
7.5	Attribute check (attributeCheck)	68
7.5.1	Usage	68
7.5.2	Properties	68
7.6	Code blocks check (blockCheck)	69
7.6.1	Usage	69
7.6.2	Properties	69
7.7	Equation check (equationCheck)	70
7.7.1	Usage	70
7.7.2	Properties	70

8	Rules file creation and edition	72
8.1	Rules file creation	72
8.2	Add a rule	74
8.3	Cancel modifications	75
8.4	Copy and past rules	75
8.5	Delete rules	75
8.6	Operations on rules properties	75
8.7	Printing rules file	76
9	Final license contract for PLC Checker	77

VAS

Chapter 1

Preamble

This document is the user guide of the PLC Checker software of Itris Automation Square.

After a short introduction, the Eclipse platform dedicated to PLC Checker is presented (installation, updates, usage,...). The rules files creation and edition and the PLC export files generation are explained in the two following chapters.

Next chapter describes the PLC Checker project creation, the way to launch the PLC Checker and how to show PLC Checker results.

An introduction to the filter and rules concepts and a description of the result file are given in the following chapter.

The last two chapters show the rules references and the link to the PLC Checker license contract.

Chapter 2

Introduction

PLC Checker analyzes a PLC code to check if it respects programming and conception rules defined in a rules file. After checking, a results file is generated. The non-respects of rules are logged in the results file. This PLC Checker version work with PLC programs of Schneider-Electric PL7-PRO, Siemens Step 7, Schneider-Electric Unity and Rockwell Automation RSLogix5000 workbenches. Step to follow for PLC Checker usage in SaaS (Software as a Service), are described in this user guide.

The PLC Checker obtains the "Grand Prix de l'Innovation 2004", organised by AUTOMATION-OPTIMIZATION EUROPE in the "meilleur logiciel d'automatisme et de gestion industrielle" category.

Chapter 3

Eclipse platform tutorial

The aim of this tutorial is to install and to get started with the Eclipse platform.

You will find more details about the installation and the necessary plugins, the definition of the workspace, workplan and project concepts and finally the most significant functions will be detailed more precisely.

3.1 Installation

This section contains the installation procedure for the Eclipse platform.

3.1.1 Hardware and software configuration needed

The Itris Automation Square workshop is integrated in the Eclipse 3.7.1 (The PLC Checker is preinstalled). This platform and the workshop specific code need the Sun java virtual machine JRE 5.0. This software configuration require a PC with following minimal characteristics:

- Microsoft Windows 2000/XP Professional/Windows 7,
- Minimal Processor frequency of 1GHz,
- 200 Mo of free disk space,
- 512 MB¹ of RAM,

3.1.2 Usage

Warning, the use of Itris Automation Square products is payable in SaaS mode (Software as a Service) and is conditioned by a commercial validation. In case of problem, please contact the commercial@automationsquare.com.

Creating a user account on the site Itris Automation Square is indispensable to use Itris Automation Square products.

In order to use Itris Automation Square products, the user has to identify himself, when asked to, by entering, in the appropriate windows, his login and password defined during his inscription on IAS website.

¹Megabyte

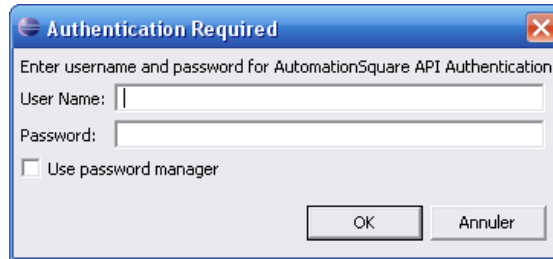


Figure 3.1: login

3.1.3 Platform installation

The installation program is available [here](#), and is named InstallPlcChecker20111121.exe. Launch the InstallPlcChecker20111121.exe in order to install the platform.

The installation program creates an entry in the windows start menu.
Detailed procedure:

- Choose installer language



Figure 3.2: Choose language

- On the welcome message, clic on **Next**



Figure 3.3: Welcome in installation program

- In order to instal the PLC Checker platform, you have to **agree the license**.

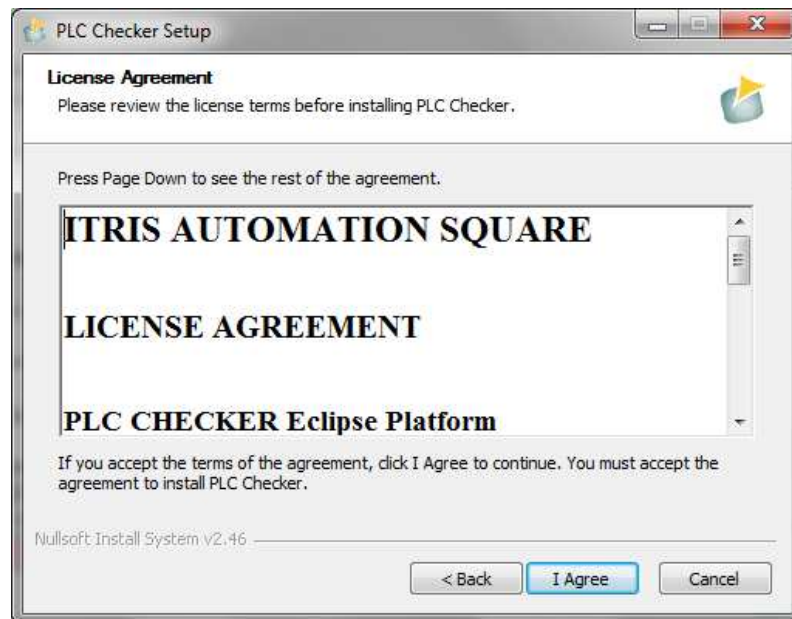


Figure 3.4: Licence

- Choose the installation directory (by default C:\IAS), then clic on **Install**

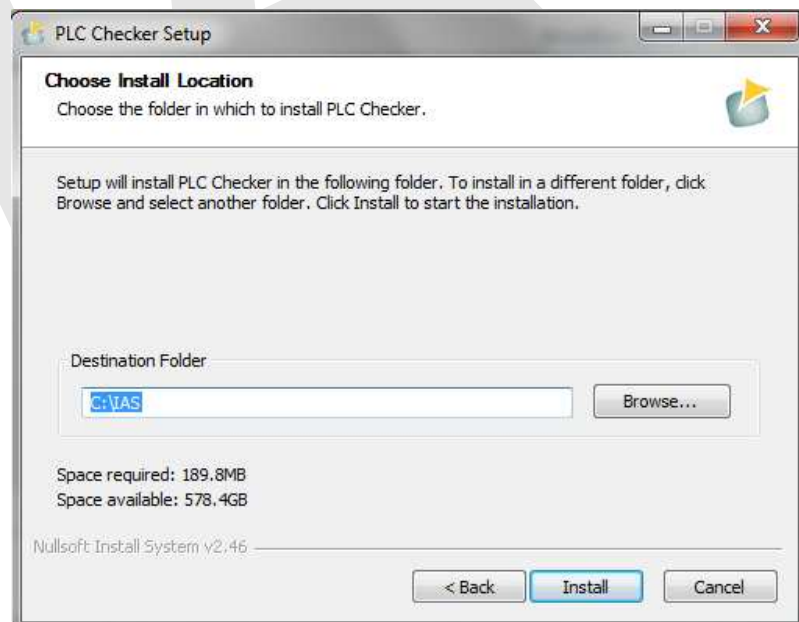


Figure 3.5: Choose the installation directory

- After installation, clic on **Finish** to close the installation program and to run the PLC Checker.

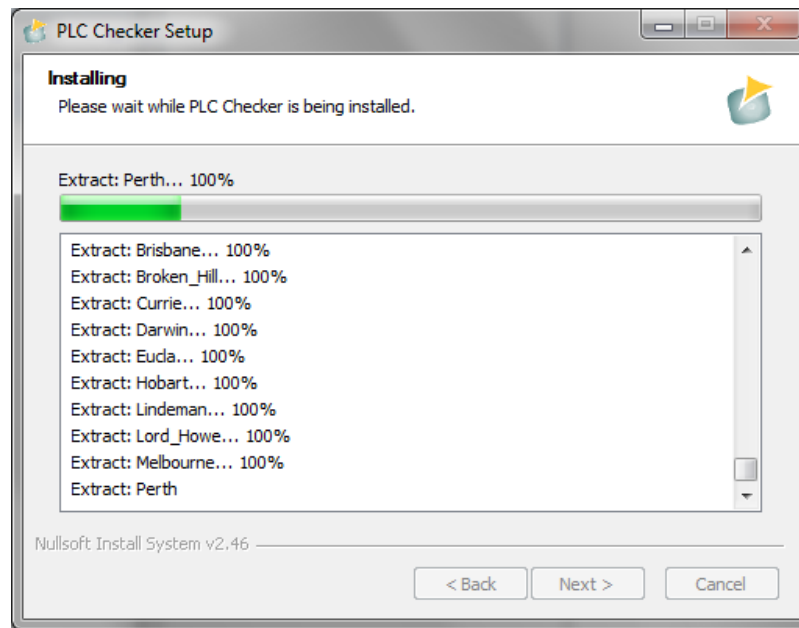


Figure 3.6: Installation

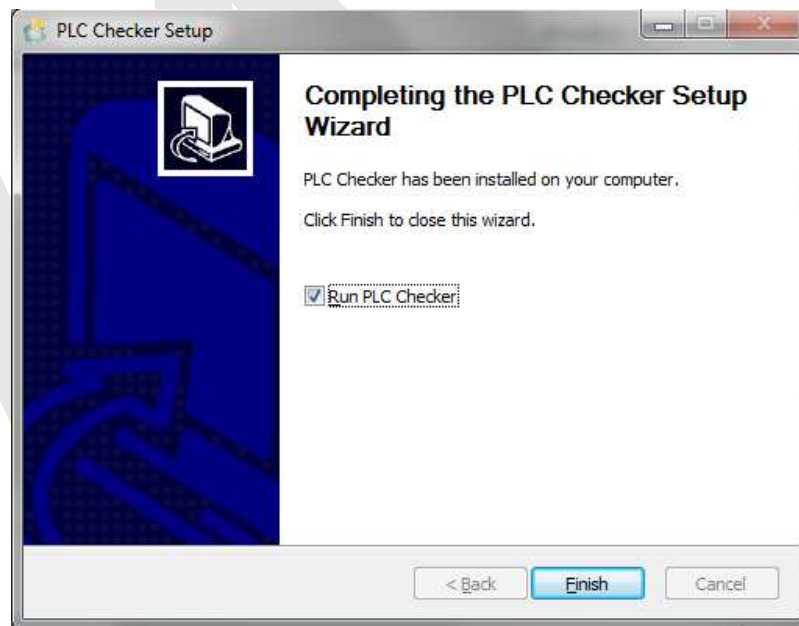


Figure 3.7: Close installation program

3.2 Software update

The search of available updates is done automatically by the platform when it starts.

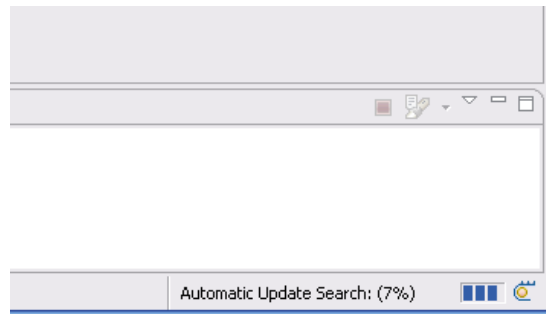


Figure 3.8: Automatic search for update

- When plugins updates are found, the platform ask you to view and install them. Clic on **Yes**.

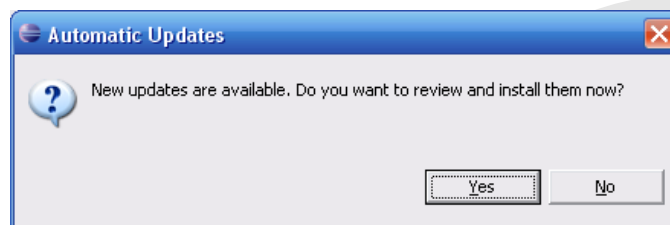


Figure 3.9: Ask for installation

- Available updates can be seen. Let all updates selected and clic on **Next**.

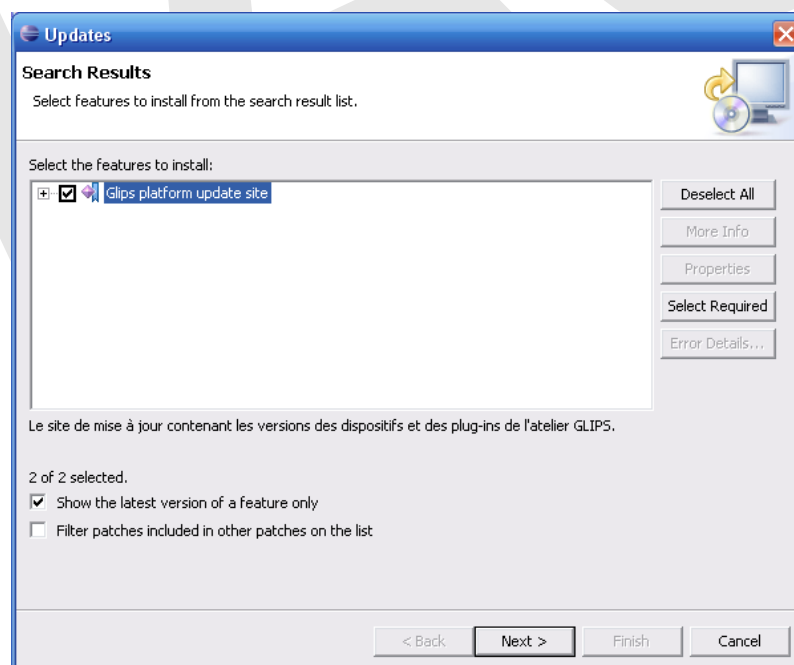


Figure 3.10: Updates selection

- **Accept license** contract, then clic on **Next**.

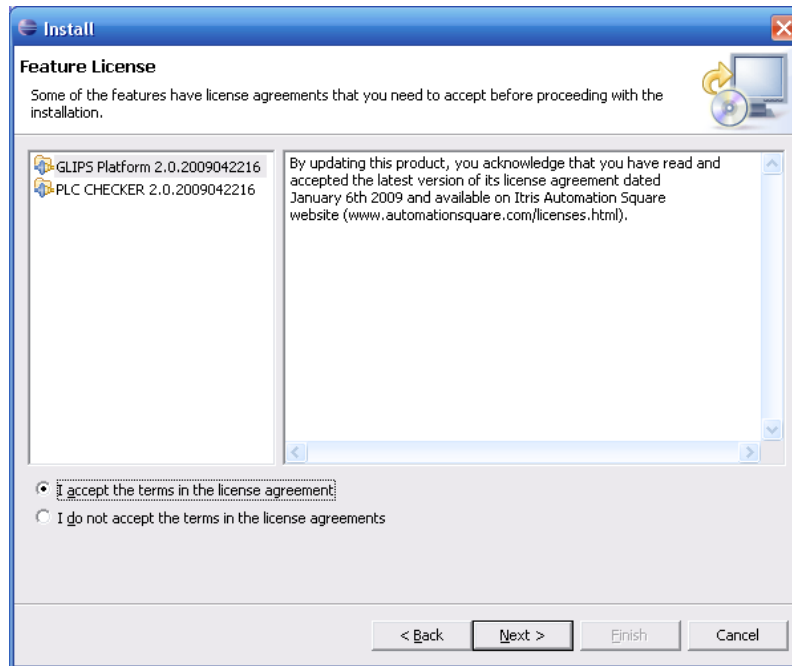


Figure 3.11: License

- The overview of updates to be installed is presented. Click on **Finish** to launch updates download.

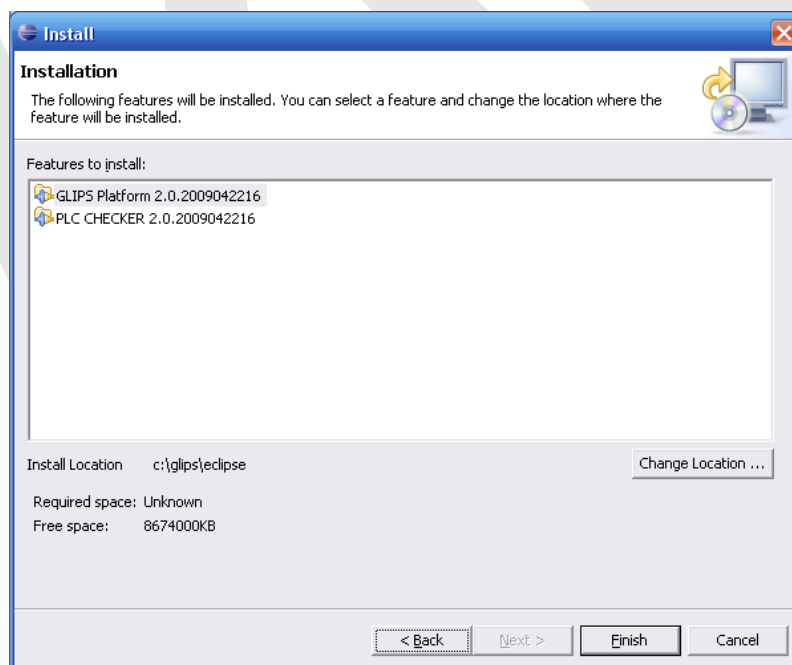


Figure 3.12: Download overview

- After download, you are asked to check the update emitter, because of unsigned updates. Click on **Install all**.

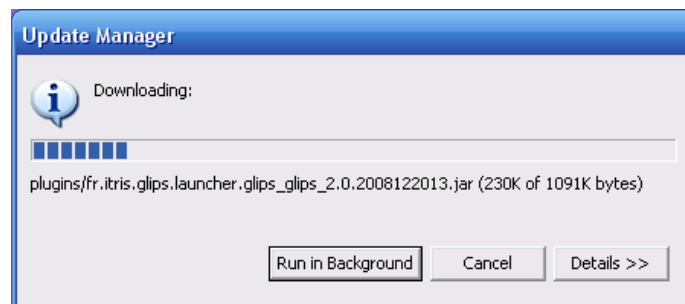


Figure 3.13: Downloading updates

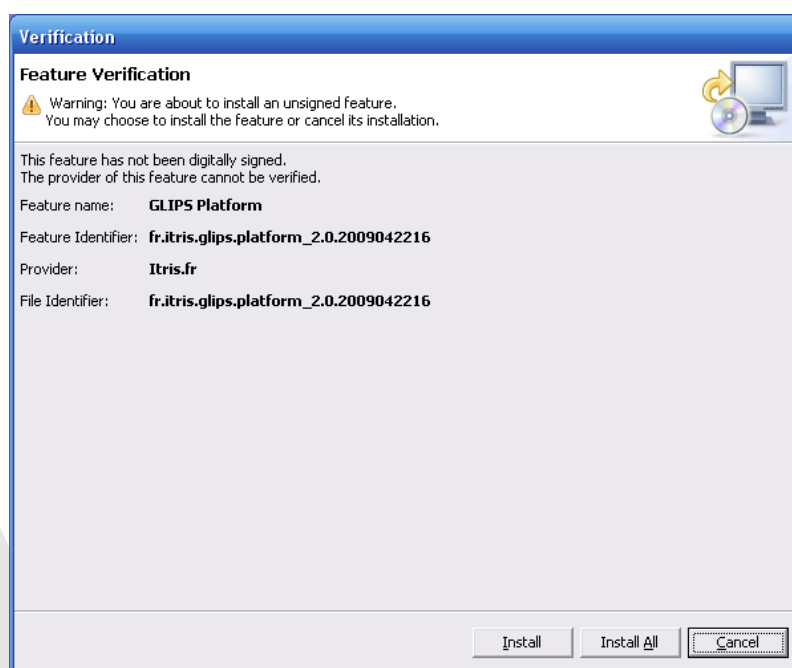


Figure 3.14: Check updates emitter

- After installation, restart the platform as asked to. Clic on **Yes**.

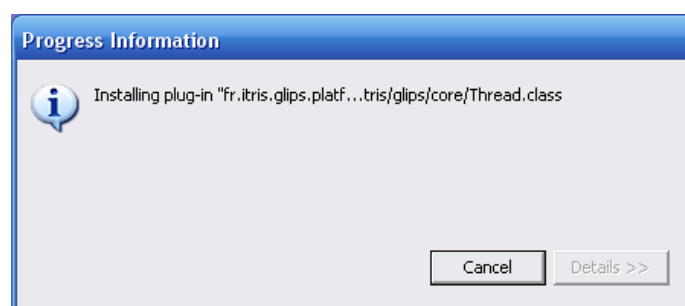


Figure 3.15: Updates installation

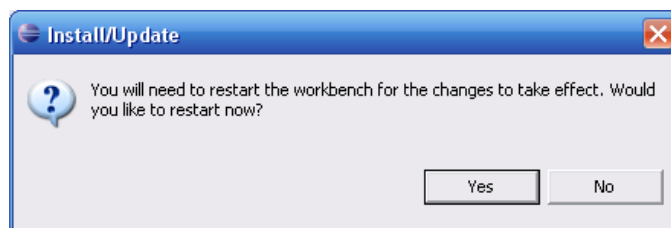


Figure 3.16: Asking to restart

3.3 Launching the platform

To launch the platform, clic on the icon from the Windows desktop or choose the Glips platform item from the Start Menu.

At the end of the installation, the platform will start automatically.

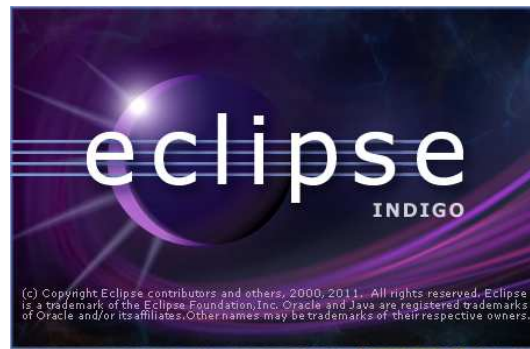


Figure 3.17: Eclipse Logo

3.3.1 Workspace

When launching the platform, the following dialog box pops up :

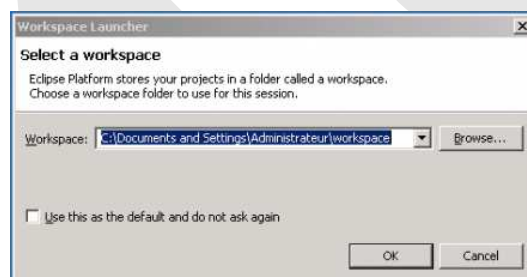


Figure 3.18: Workspace selection

The workspace is a hard disk drive directory reserved for Eclipse needs. In this specified folder, Eclipse builds up another folder called `.metadata` containing all the data it needs to work. Note that it is important :

1. To split the Eclipse workspace from your current work folders
2. Not to use the Windows explorer to generate files in the workspace (for this purpose use the import/export function or file drag'n drop function)
3. To have only one workspace

The `.metadata` contains the Eclipse project list.

Once the folder chosen, you will get rid of the start question on next starts of the platform by ticking the option of the dialog box.

3.3.2 Identification

When you are asked, identify yourself by typing in the window, your login and password that you obtain when you register on our website at register page .

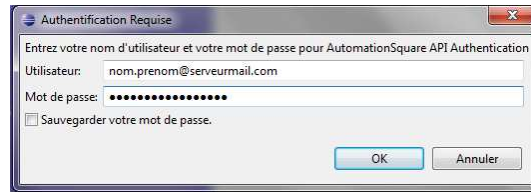


Figure 3.19: login

3.3.3 Work bench

When first starting the platform, the following welcome screen is displayed. You may follow the Eclipse tutorials. You also may quit this screen by clicking on "go to the work bench" or closing the inner window.

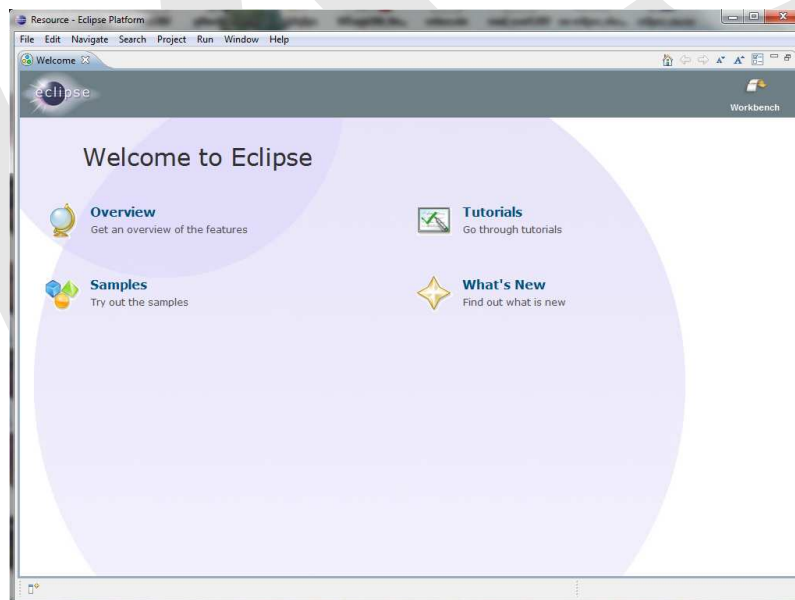


Figure 3.20: Eclipse welcome page

The Eclipse work bench includes several windows. The default layout is as shown on the following picture :

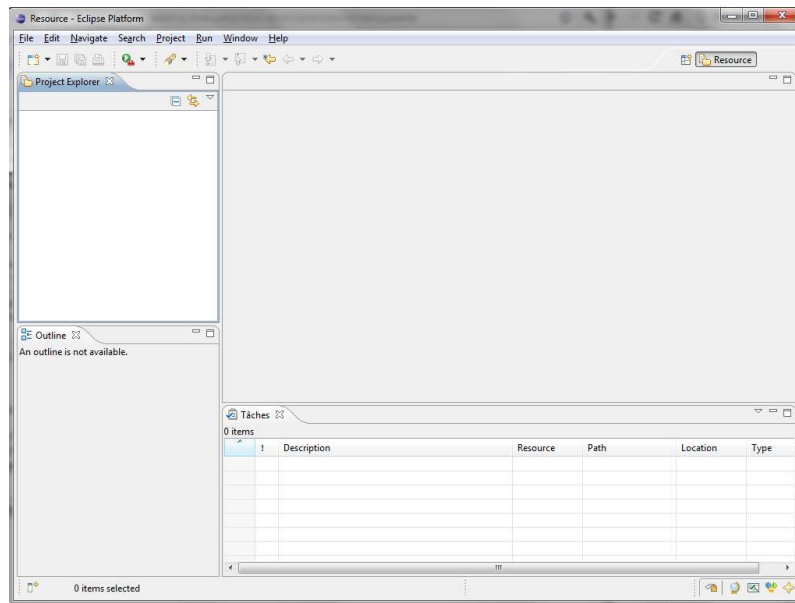


Figure 3.21: Eclipse default layout

On the left part is the explorer. You may view and browse your working space.

On the right part, the big grey window displays the used editors when opening files of the work space.

At the bottom left, a structured overview of the current editor is displayed. It is empty in the present case.

At the bottom is the 'to do' window in which you will find tasks to do. Eclipse allows you to develop and add more tasks in order to inevitably proceed everything without omission.

Warning: updating platform is done automatically each time you start it. If the interface does not seem to answer, make sure it is not an Update.

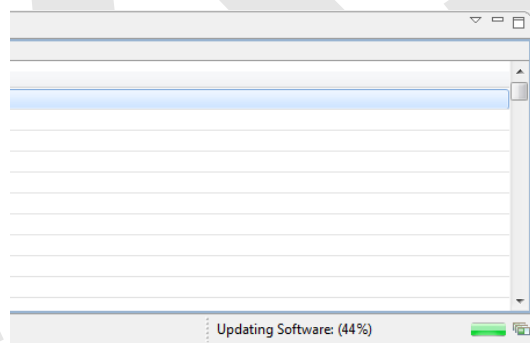


Figure 3.22: updating

3.4 Import/Export files into/from the project

File import must be done only using the Eclipse feature. There is no way using the Microsoft file browser. Importing or exporting data can be completed in different ways in the workspace. The drag'n drop and use of import features are described hereunder.

3.4.1 Drag'N Drop

The following snapshots show the different steps to import files into the workspace using the drag'n drop way. A file browser must be open and you only need to drag the desired files and drop them to the workspace. This operation does not erase the files from their original location.

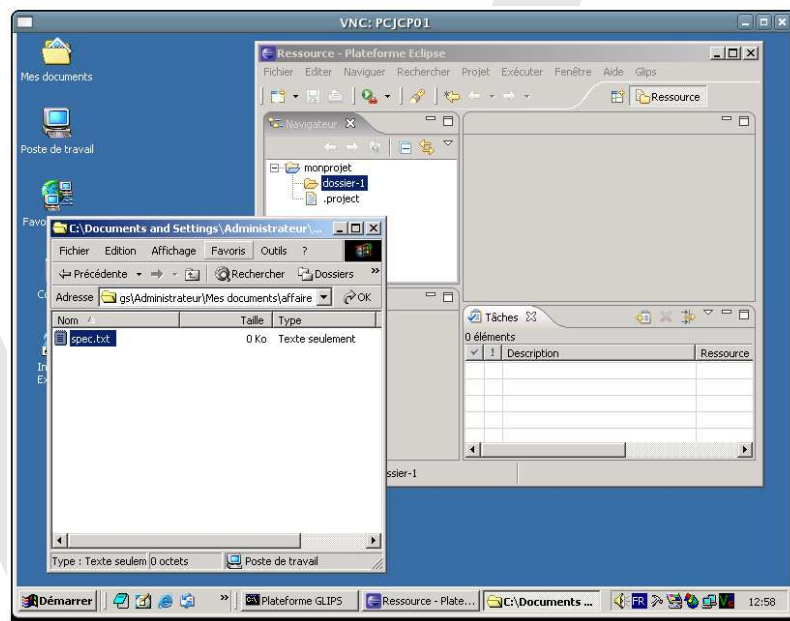


Figure 3.23: Drag'n Drop files

3.4.2 Importing files

The import feature using the dialog boxes is more complex. It is more suitable for importing only few files among others of a folder; selecting them becomes easier.

In the following example, you wish to import files into myfolder. myfolder must be selected, right-click and choose 'Import'

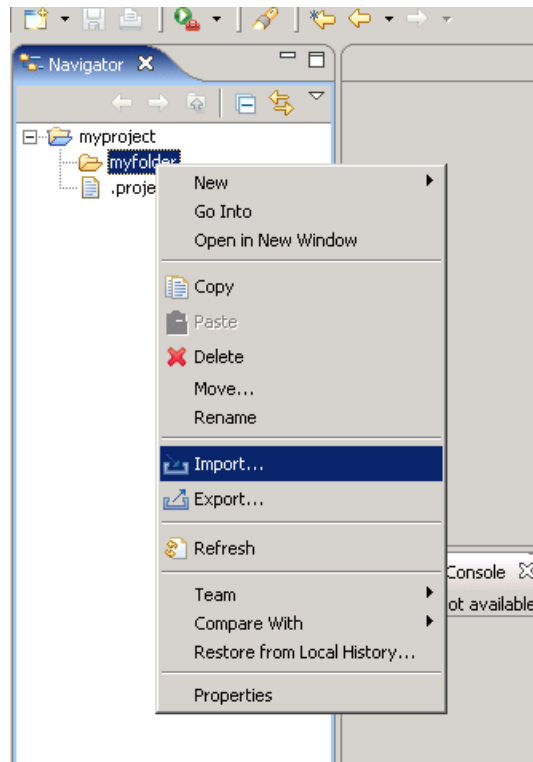


Figure 3.24: Contextual menu - Import

Among all the alternatives, importing files from the file system is the most interesting.

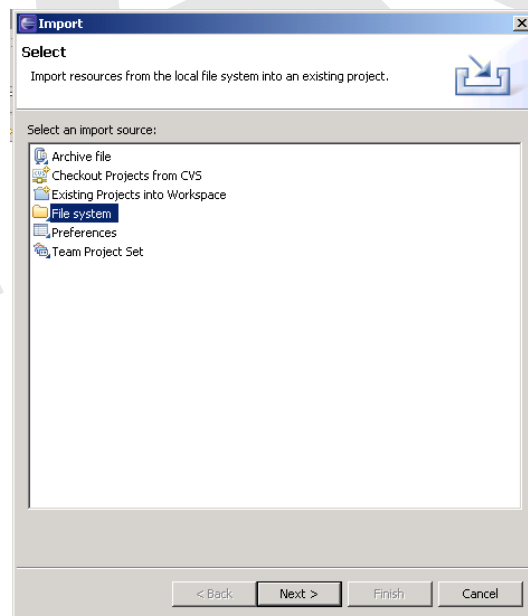


Figure 3.25: Select file system source for import

You must now select the hard disk directory from which the files will be imported.

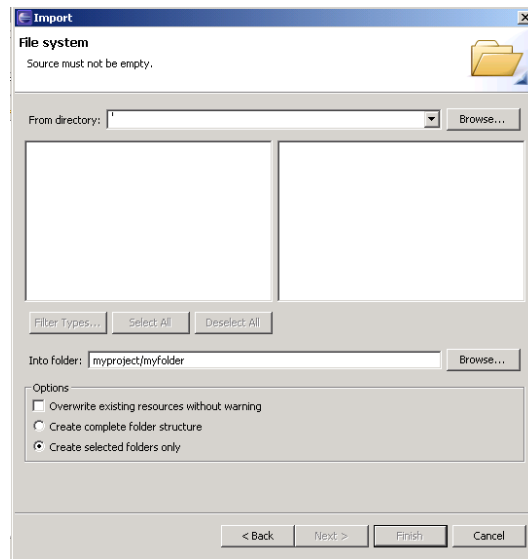


Figure 3.26: Select directory for import

Click on 'Browse'. A folder selection box pops up.

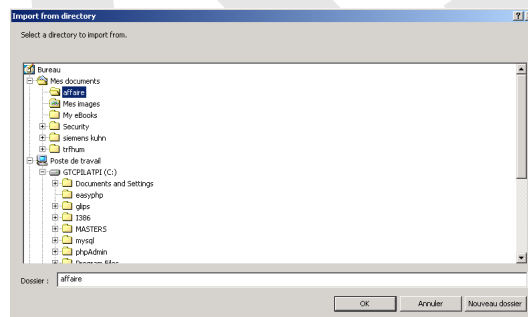


Figure 3.27: Browse directory to select one

Once the source folder is chosen, confirming displays the list of importable files. Select now the files to be imported, in our case we choose 'spec².txt'.

²specification

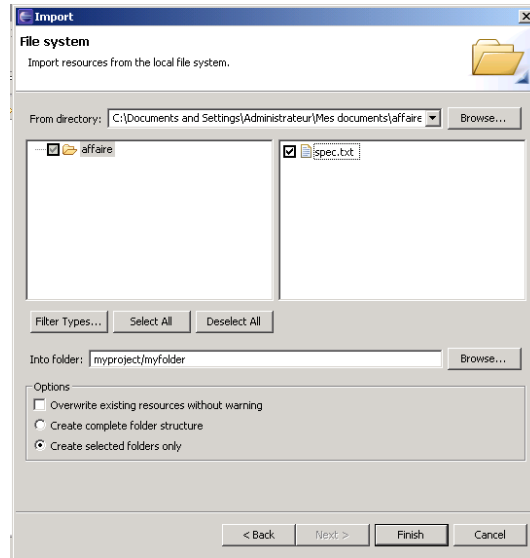


Figure 3.28: In the directory, select the file(s) to be imported

After confirming the desired file is imported into the workspace.

3.5 Comparing files

Comparing files in the Eclipse environment is a very easy operation.

Select first the both files to be compared.

The popup menu, obtained by a right clic, allows you to perform a crossed comparison between selected files (menu: compare / with each other).

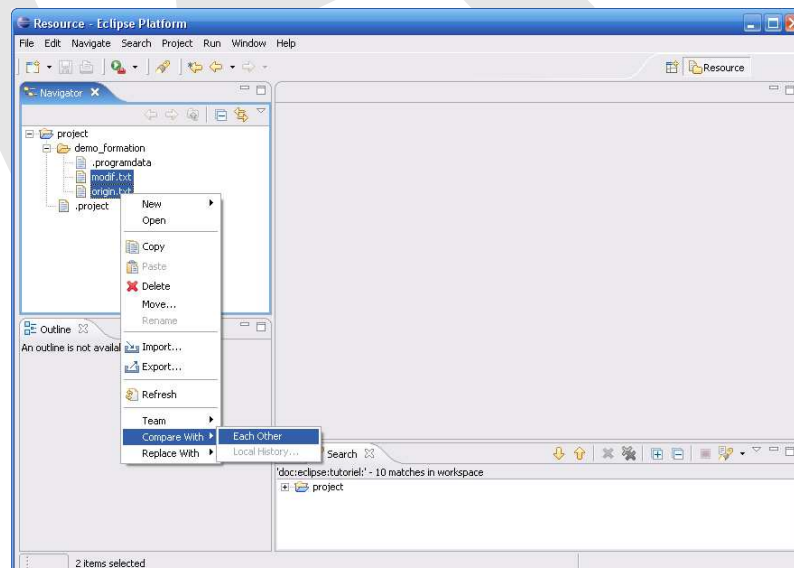


Figure 3.29: Launch files comparison

Selecting the menu input allows to open the difference editor :

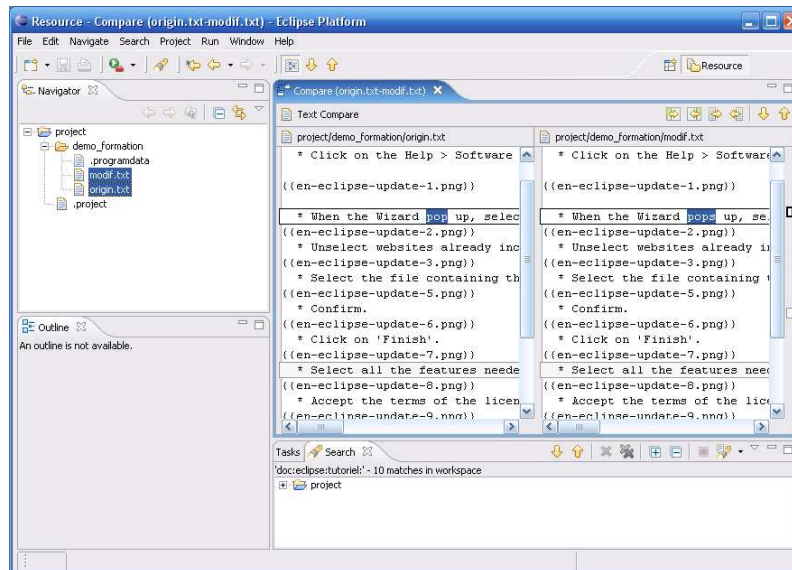


Figure 3.30: Look for differences

You can shift between differences by clicking the small white squares on the right side of the window or by using up and down arrows on the top right side.

To exit the comparison mode, only close the main window.

3.6 Local history

Eclipse provides a local history feature for all text files (not only *.txt) that belong to a project.

Traces of iterative modifications of text files are kept.

At Anytime, the user is able to check the differences between the current version and the former ones and get back to a previous version quickly and easily.

3.6.1 Compare with a older version

Only select one file in the Eclipse environment. Select 'Compare > Local history' with the popup menu.

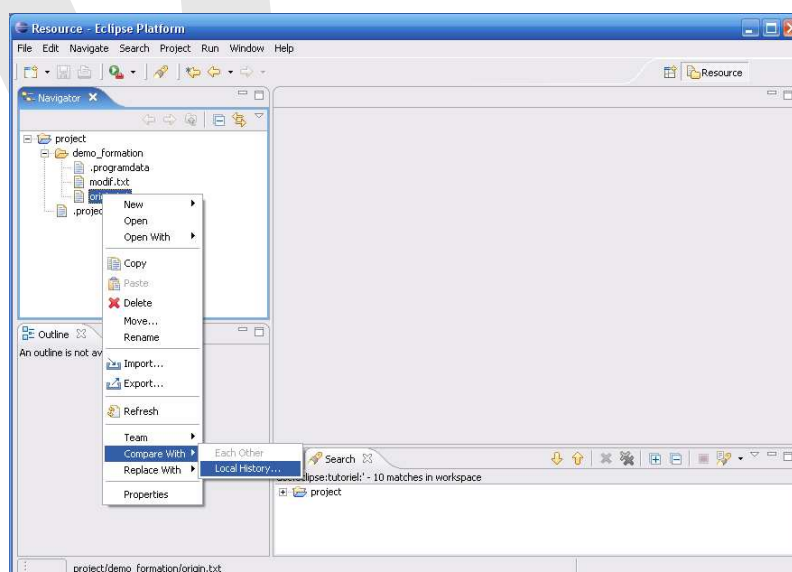


Figure 3.31: Launch local history comparison

The windows that pops up looks pretty much like the one to compare files at the top which is the local history of the file.

Successively selecting versions gives you the differences between the current version and the selected one.

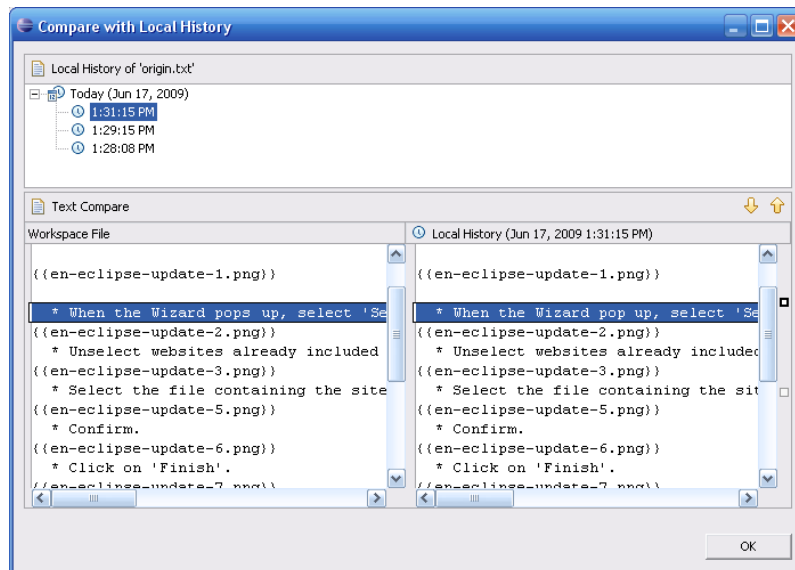


Figure 3.32: Local history comparison

3.6.2 Replacing by an older version

In the popup menu, choosing 'Replace by local history', replacing the current version by an older version becomes possible, at the same time viewing the differences.

Chapter 4

PLC Checker project creation and usage

The synthetic procedure for project creation, PLC Checker usage and results visualization is the following:

- **Preamble:**
 1. Create an user account on Itris Automation Square [website](#)
 2. Buy PLC Checker rights to Itris Automation Square commercial@automationsquare.com
 3. Obtain rules file from your client or your quality service or create and edit your own rules file.
 4. Generate your PLC application Export file(s) from your PLC workshop.
- **Create project:**
 1. Create a PLC Checker project on the [Eclipse plateforme](#)
 - (a) Name and describe your project on the [Eclipse plateforme](#)
 - (b) - Associate a right to your project on the [Eclipse plateforme](#)
 2. Create a PLC Checker project on the Itris Automation Square [website](#)
 - (a) Name and describe your project on the website (step 1)
 - (b) Associate a right to your project on the website (step 4)
- **Execute PLC Checker:**
 - **Method 1** : from the [Eclipse platform](#) :
 1. Create an PLC Checker project in your workspace of the Eclipse platform, that gets the contents and the characteristics of the PLC Checker project previously created on the website.
 2. Import PLC application export file(s) in this project folder on your Eclipse workspace.
 3. Import the rules file in this project folder on your Eclipse workspace.
 4. Execute PLC Checker from your Eclipse workspace.
 - **Method 2** : from the [website](#):
 1. Upload PLC application export file(s) to the server from the website (step 2).
 2. Upload rule file to the server from the website (step 3).
 3. Execute PLC Checker on the server from the website (step 5).
- **View results on the [Eclipse platform](#):**
 1. If you have choose method 2, you need to create an PLC Checker project in your workspace of the Eclipse platform, that gets the contents and the characteristics of the PLC Checker project previously created on the website (see method 1).
 2. View results file (*.gqre) on Eclipse platform.

4.1 PLC Checker project creation

4.1.1 Default PLC Checker project creation with a wizard on Itris Automation Square website

- Choose the PLC workshop type (PL7-Pro, Unity, Step 7, RSLogix5000) and the kind of verification (Single Run / Unlimited), then clic on button "Launch Wizard".

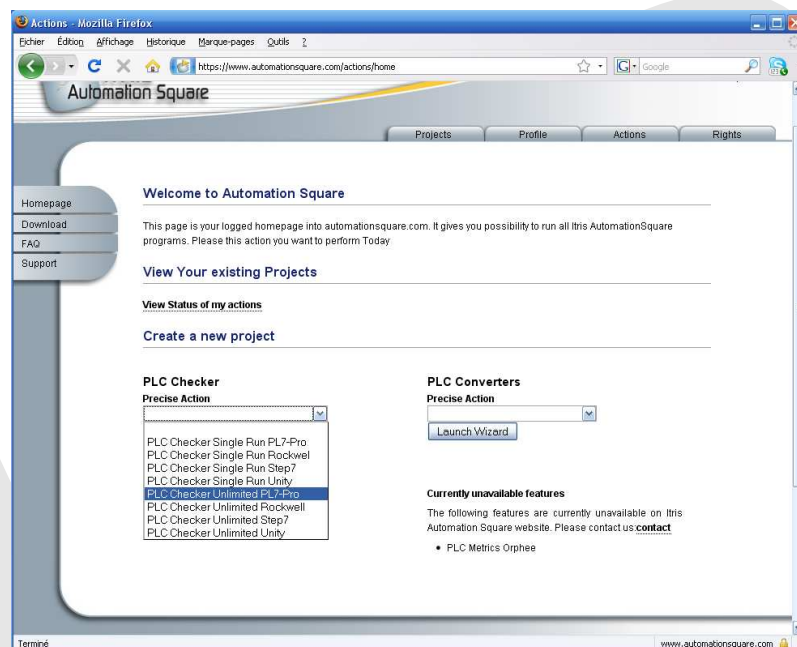
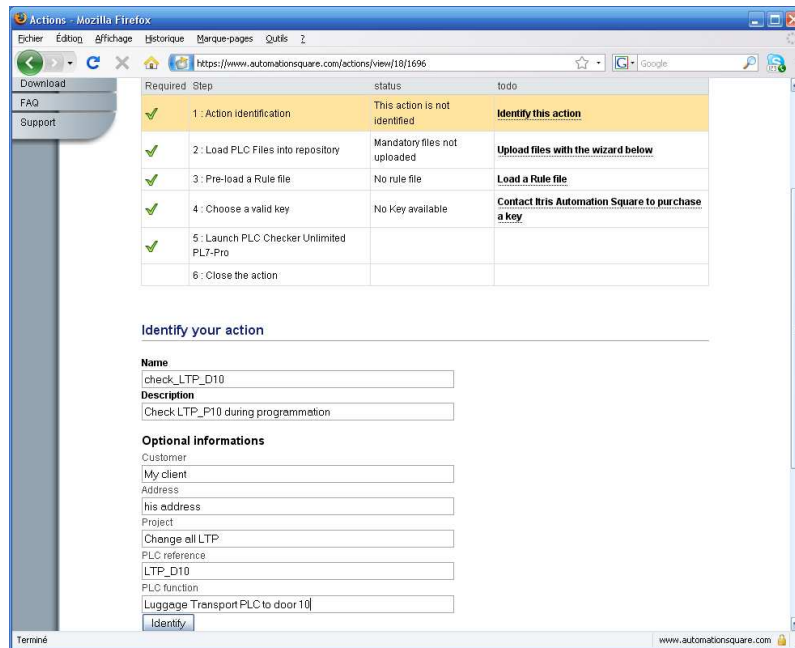


Figure 4.1: Choose PLC Checker functionality

- Name the project from website (step 1):
 1. Give a name and a description of the project.
 2. Some complementary informations can be specified for the project. This can be modified later by using link "Change your identification".
 3. Clic on button "Identify".



Required	Step	status	todo
✓	1 : Action identification	This action is not identified	Identify this action
✓	2 : Load PLC Files into repository	Mandatory files not uploaded	Upload files with the wizard below
✓	3 : Pre-load a Rule file	No rule file	Load a Rule file
✓	4 : Choose a valid key	No Key available	Contact Itris Automation Square to purchase a key
✓	5 : Launch PLC Checker Unlimited PL7-Pro		
	6 : Close the action		

Identify your action

Name
check_LTP_D10

Description
Check LTP_P10 during programming

Optional informations

Customer
My client

Address
his address

Project
Change all LTP

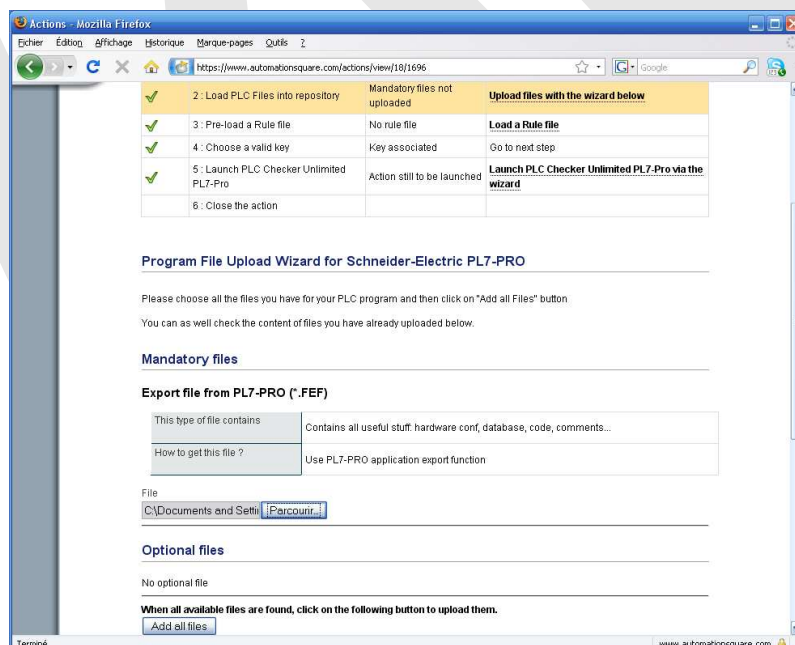
PLC reference
LTP_D10

PLC function
Luggage Transport PLC to door 10

[Identify](#)

Figure 4.2: Project description

- Download the PLC application export file(s) on the server from the website (step 2).
 - If you are not at step 2, clic on the corresponding link **"Upload files with the wizard below"**.
 - Search on your disk the mandatory and optional files using the **"File"** field
 - Clic on the **"Add all files"** button to upload files on the IAS server.



Required	Step	status	todo
✓	2 : Load PLC Files into repository	Mandatory files not uploaded	Upload files with the wizard below
✓	3 : Pre-load a Rule file	No rule file	Load a Rule file
✓	4 : Choose a valid key	Key associated	Go to next step
✓	5 : Launch PLC Checker Unlimited PL7-Pro	Action still to be launched	Launch PLC Checker Unlimited PL7-Pro via the wizard
	6 : Close the action		

Program File Upload Wizard for Schneider-Electric PL7-PRO

Please choose all the files you have for your PLC program and then click on "Add all Files" button.
You can as well check the content of files you have already uploaded below.

Mandatory files

Export file from PL7-PRO (.FEF)

This type of file contains
Contains all useful stuff: hardware conf, database, code, comments...

How to get this file ?
Use PL7-PRO application export function

File
C:\Documents and Settings\Perceur\...

Optional files

No optional file

When all available files are found, click on the following button to upload them.

[Add all files](#)

Figure 4.3: Select PLC application export file(s)

- Download the rules file on the server from the website (step 3).
 - If you are not at step 2, clic on the corresponding link **"Load a rules file"**.

2. Search on your disk the rules file using the **"File"** field
3. Clic on the **"Add all files"** button to upload files on the IAS server.

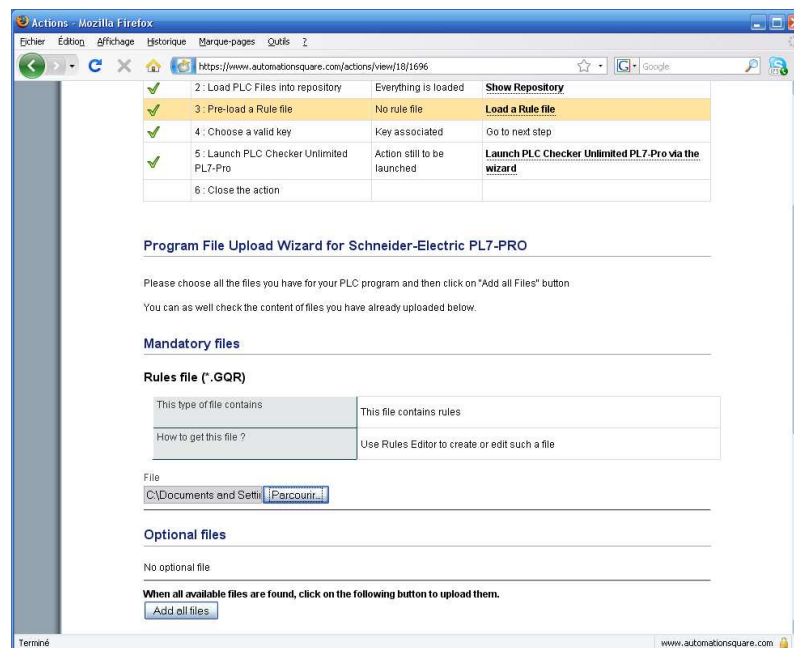


Figure 4.4: Select rules file

- Associate a right to the project from the website (step 4).

If you do not already have right for this action, please use right order interface or contact the commercial@automationsquare.com.

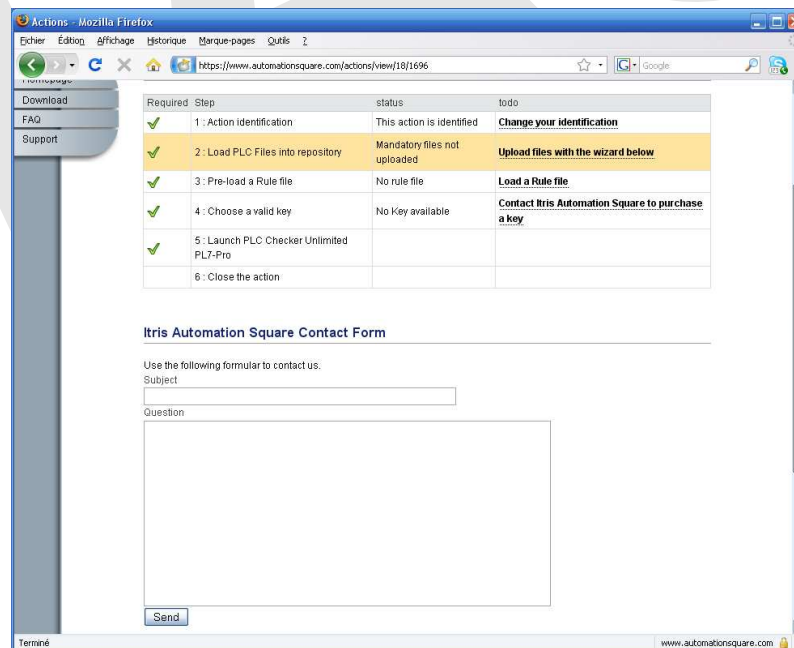


Figure 4.5: Ordering a right

If you have right for this action, select a right to be associated, and clic on the **"Associate"** button.

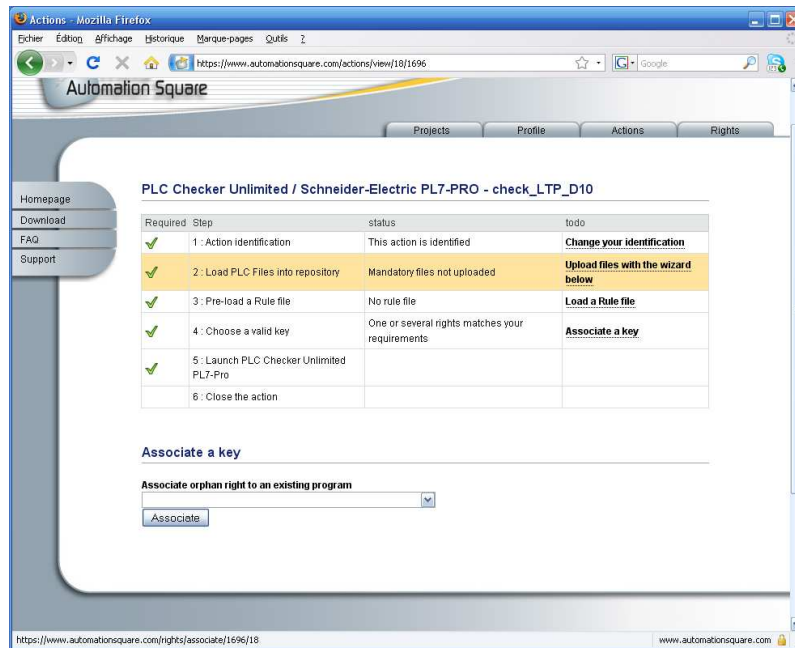


Figure 4.6: Associate a key

Note your program code, it will be useful for IAS project creation on the Eclipse platform of the PLC Checker.

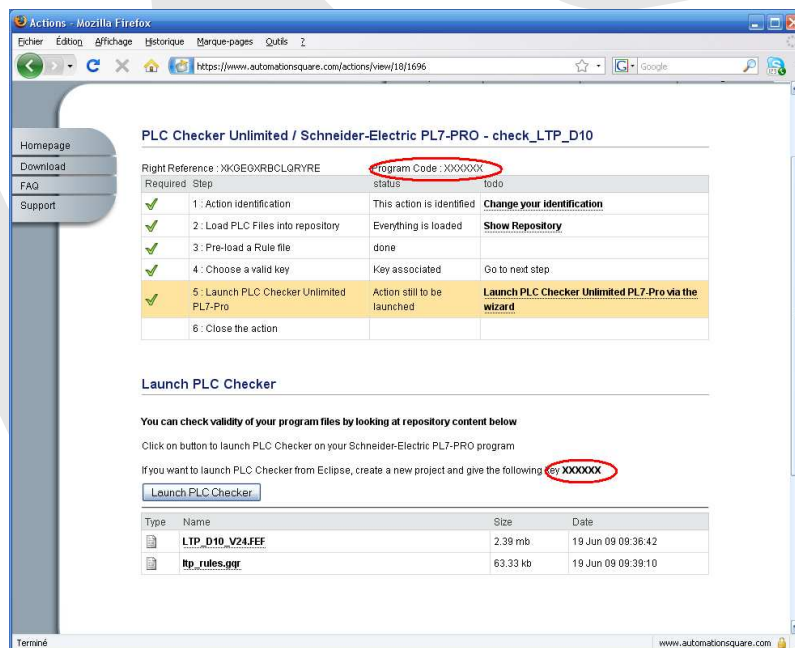


Figure 4.7: Copy program code

4.1.2 Creation on IAS with wizard (PLC Checker project to be defined).

- Access to "My projects" page (top left corner).



Figure 4.8: Access to the projects page

- Create a new project to contain the programs or select an existing project.

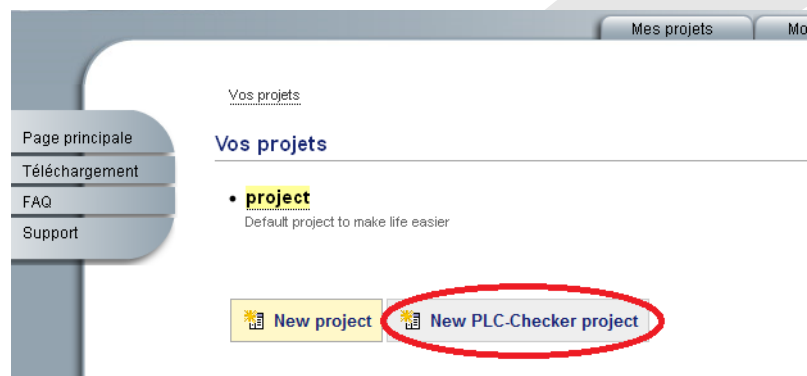


Figure 4.9: New project creation

- This page allows the creation of new project:
 1. Give a name and a description of the project.
 2. Some complementary informations can be specified for the project (name, customer address and rules file ...). Those project informations are a default values for all corresponding program's data.
 3. Enter the program characteristics (name, description, PLC type), Associate a right to the program from the box "PLC Checker Right" (one right / program).
 - If you do not already have right for this action, please use right order interface or contact the commercial@automationsquare.com.
 - If you have right for this action, select a right to be associated from the box "PLC Checker Right".
 4. A Additional information can be entered for the program (name, customer address, rules file...). those Informations can override equivalent global project information for this particular program.
 5. Create an other(s) new program(s).
 6. To finish Clic on button "Create Project".

Project Checker creation

This page is intended to create a new project to manage quality using PLC Checker.

Project creation

Project characteristics

Name: (1)

Description:

Project global data

The content of these fields is a default value for corresponding programs data.

Customer name:

Project Site address: (2)

Project site city:

Rules set:

Information for program #1

Name: (3)

PLC codename: (4)

PLC function:

Drop PLC Checker: (5)

Customer name:

PLC Site address:

PLC site city:

Rules set:

Information for program #2

Name:

PLC codename:

PLC function:

Drop PLC Checker:

Customer name:

PLC Site address:

PLC site city:

Rules set:

Information for program #3

Name:

PLC codename:

PLC function:

Drop PLC Checker:

Customer name:

PLC Site address:

PLC site city:

Rules set:

The following information is used to identify precisely and contextually the PLC and his associated code

(6)

Figure 4.10: New project description

- Note your 6 chars program code, it will be useful for PLC Checher project creation on the Eclipse platform of the PLC Checker.

My_project / project details

Project : My_project

Owner:

Creation: 2011-12-06 11:26:39

Parameters:

- Open
- Program count: 2

Related Programs

- Detailed characteristics of program : My_program1
SUPPLK
- No result available for this program.
- Detailed characteristics of program : My_program1
PKSNOA
- No result available for this program.

Figure 4.11: Copy program code

- The detailed specifications of a each program can be changed later by clicking on button "View".

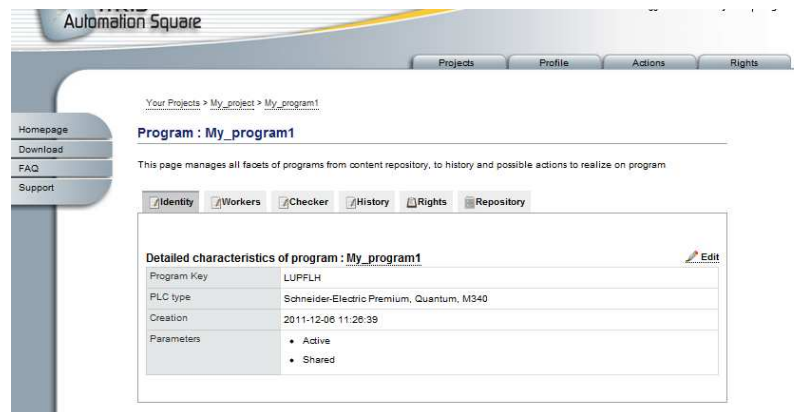


Figure 4.12: Detailed specifications

4.1.3 Creation on IAS without wizard.

- Access to "My projects" page (top left corner).



Figure 4.13: Access to the projects page

- Create a new project to contain the programs or select an existing project.

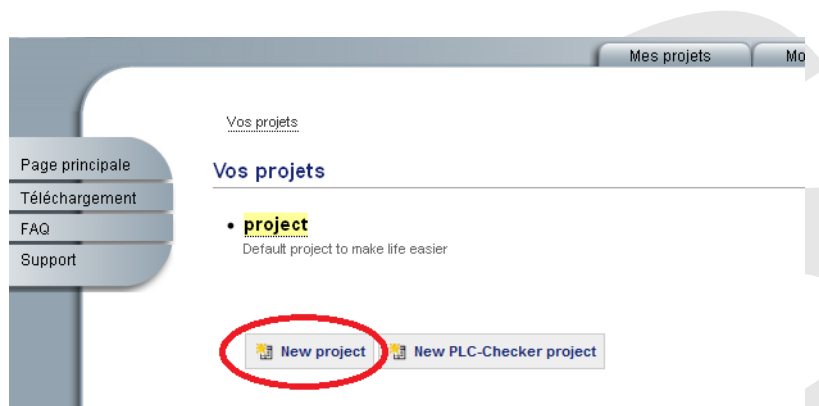


Figure 4.14: New project creation

- Enter the project name and description and validate with the "Add" button.

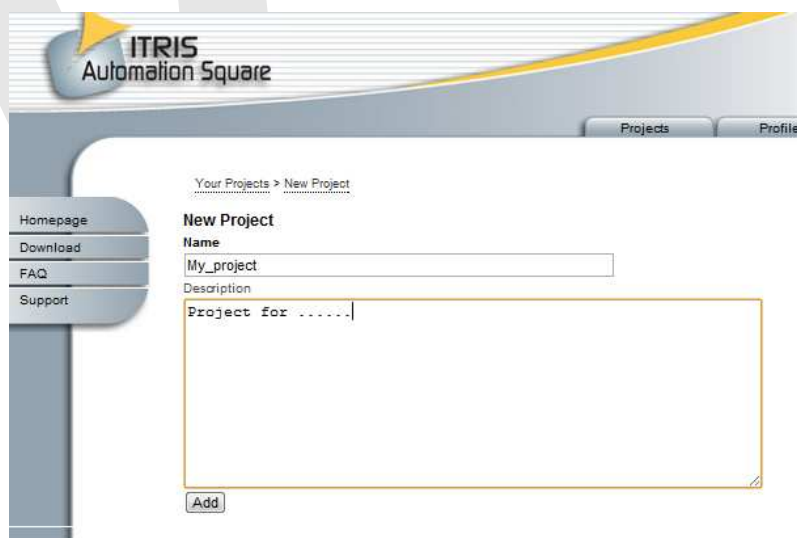


Figure 4.15: New project description

- Select the project that will contain the programs.

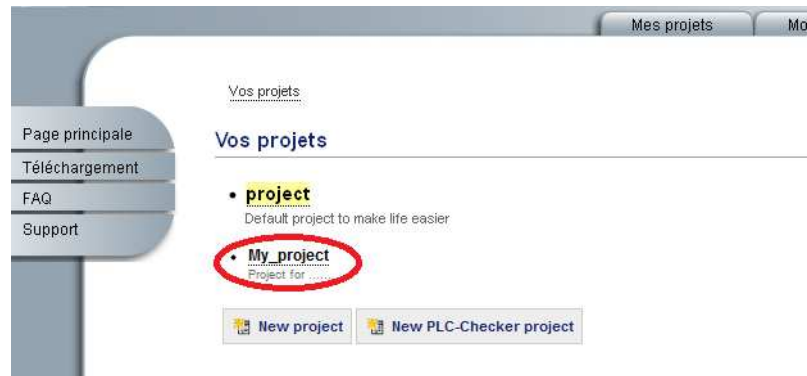


Figure 4.16: Project selection

- Create a new program.

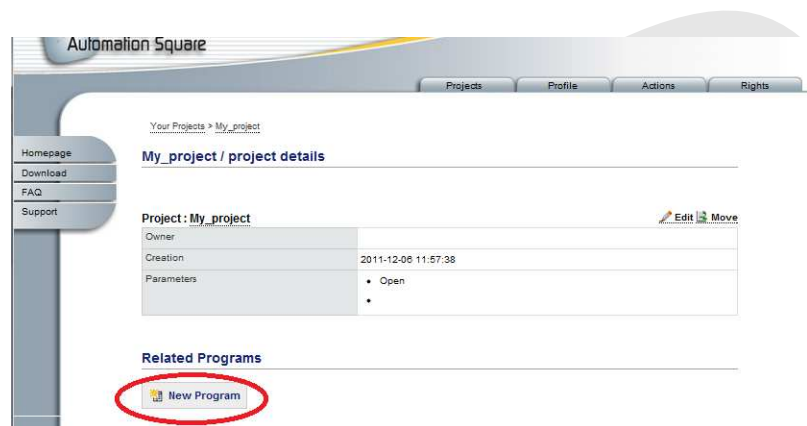


Figure 4.17: New program creation

- Enter the program name and description and the PLC type and validate with the "Add" button.

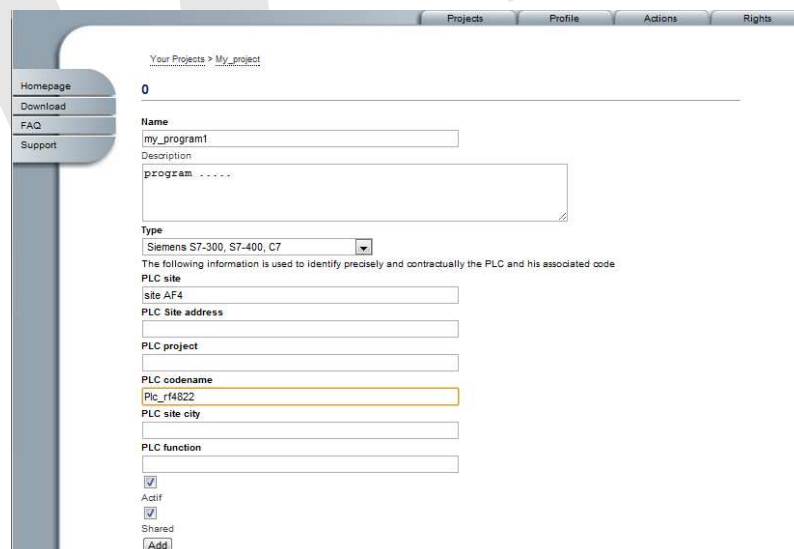


Figure 4.18: New program description

- Select the program.

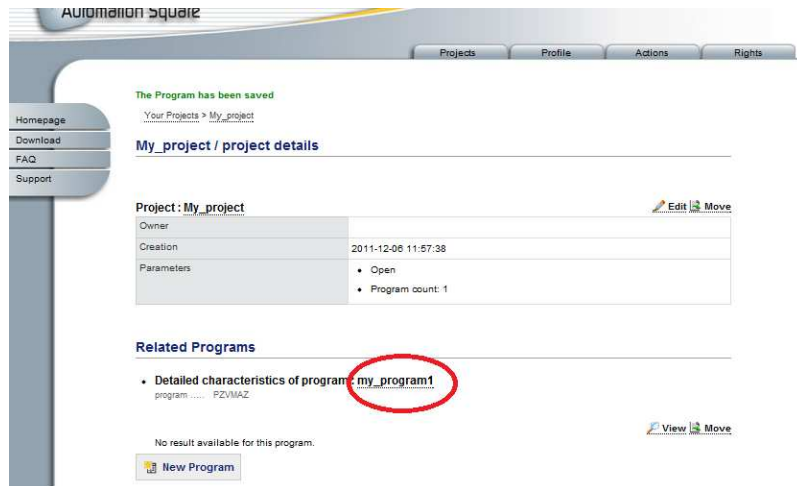


Figure 4.19: Program selection

- Associate a right to the program.
 - If you do not already have right for this action, please use right order interface or contact the commercial@automationsquare.com.
 - If you have right for this action, select a right to be associated, using the "Right" tab, and clic on the "Associate the right to the program" button.



Figure 4.20: Rights page

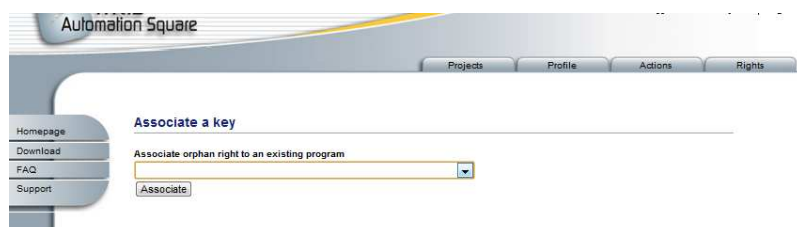


Figure 4.21: Right selection

- Note the 6 chars program code, it will be useful for IAS project creation on the Eclipse platform of the PLC Checker.

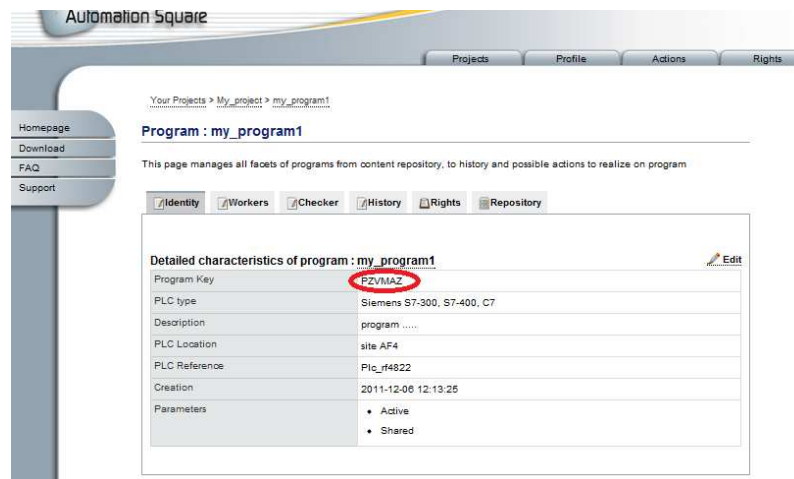


Figure 4.22: Copy program code

4.1.4 New PLC Checker project creation on Eclipse platforme

- Create a PLC Checker project in the Eclipse platform, then clic on **Next** button

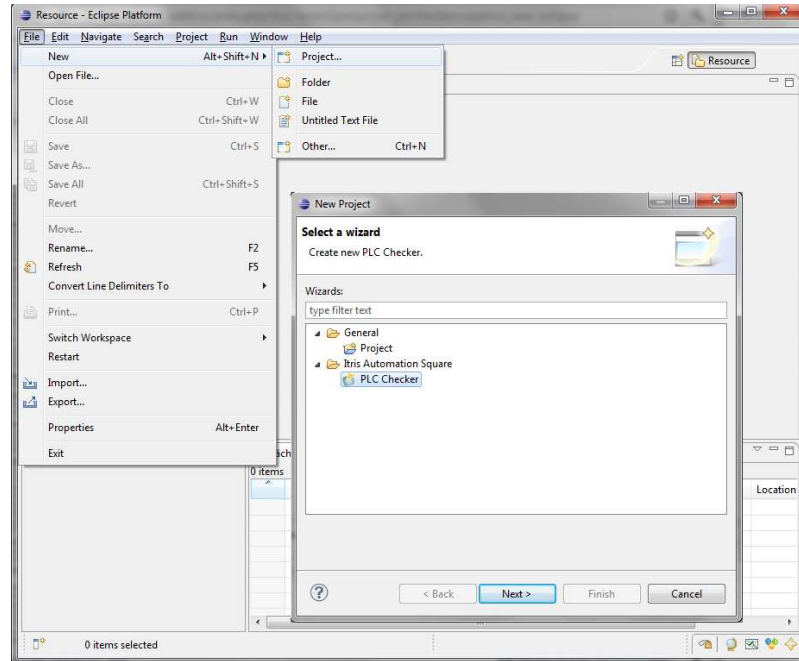


Figure 4.23: New project

- Choose the plc type (PL7-Pro, Unity, Step 7, RSLogix5000) in the **New PLC Program**
- Enter the project name, description and the PLC type.
- Associate a right to the program.
 - If you do not already have right for this action, please use right order interface or contact the commercial@automationsquare.com.
 - If you have right for this action, select a right to be associated, please select one in the **IAS Right**.
- Click on **Finish** button. The platform creates your workspace directory with the name of the project defined before.

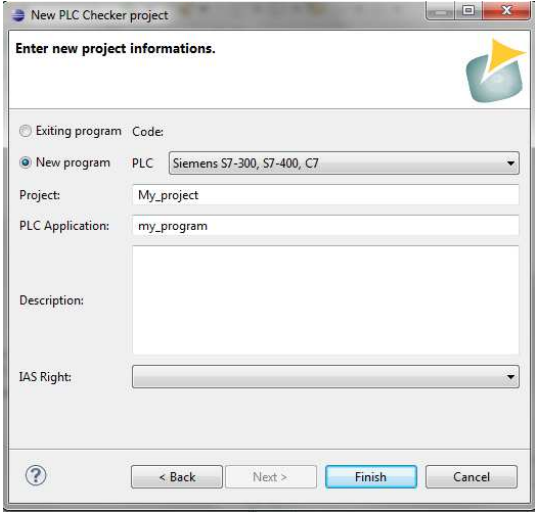


Figure 4.24: New Plc checher project

4.1.5 Existing PLC Checker project creation on Eclipse platforme

- Create a PLC Checker project in the Eclipse platform, that gets the contain and characteristics of the project previously created on the website

On Eclipse platform, select a new the PLC Checker project then clic on **Next** button.

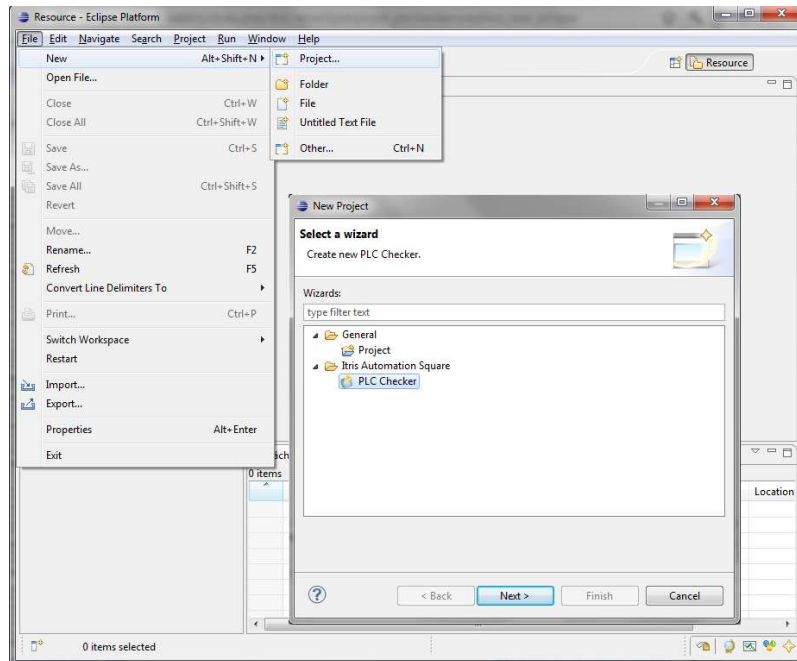


Figure 4.25: New project

Enter the program code of your project in the **Code** field. Clic on the **Finish** button. The platform create on your workspace a folder named like the project defined on the website. The platform also copy in that folder the files contained in you project on the IAS server.

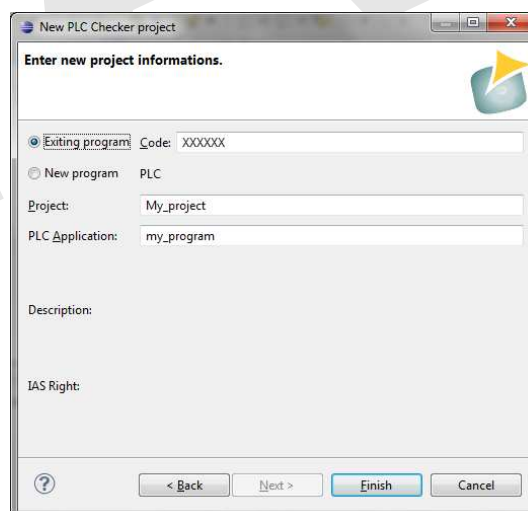


Figure 4.26: Enter program code

4.2 Run PLC Checker

4.2.1 Run PLC Checker from an Eclipse platform :

- Import application file(s) in your workspace.
 1. Use the file **explorer** of your OS¹ to **copy** the application export file(s) to be used.
 2. **Past** this(those) file(s) in your project folder on the **Navigator** view of Eclipse platform.
- Import rules file in your workspace.
 1. Use the file **explorer** of your OS¹ to **copy** the rules file to be used.
 2. **Past** this file in your project folder on the **Navigator** view of Eclipse platform.

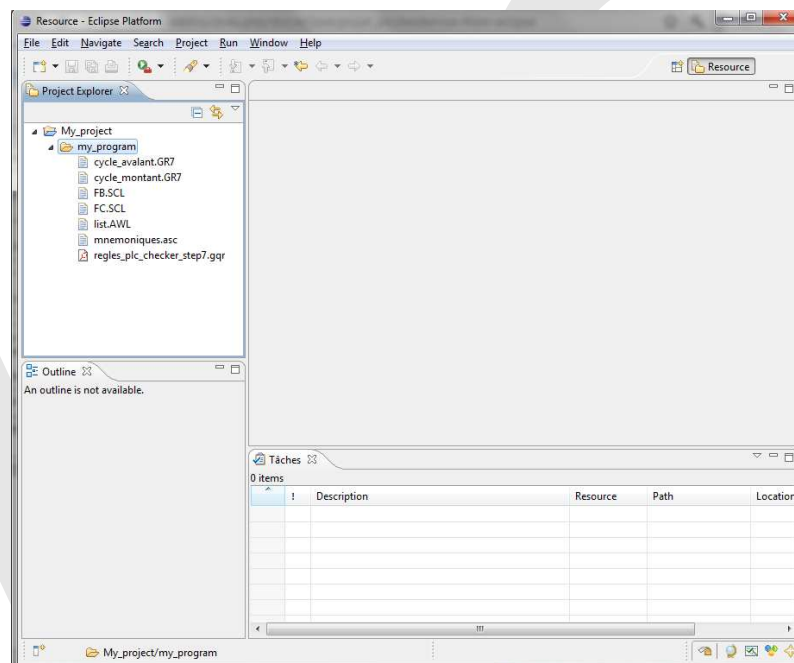


Figure 4.27: Project folder view by Eclipse Navigator

- Run PLC Checker from the Eclipse **Navigator**.
 1. **Select the application export file(s)** (*.FEF pour PL7PRO, *.ASC, *.AWL, *.GR7 et *.SCL pour STEP7, *.XEF pour UNITY-PRO, *.L5K pour RSLogix5000) to check and the **rules file** (*.GQR) to be used to.
 2. Open **right clic** contextual menu
 3. Launch the analyze using the **GLIPS tools / Run PLC Checker for ...**

¹Operating System

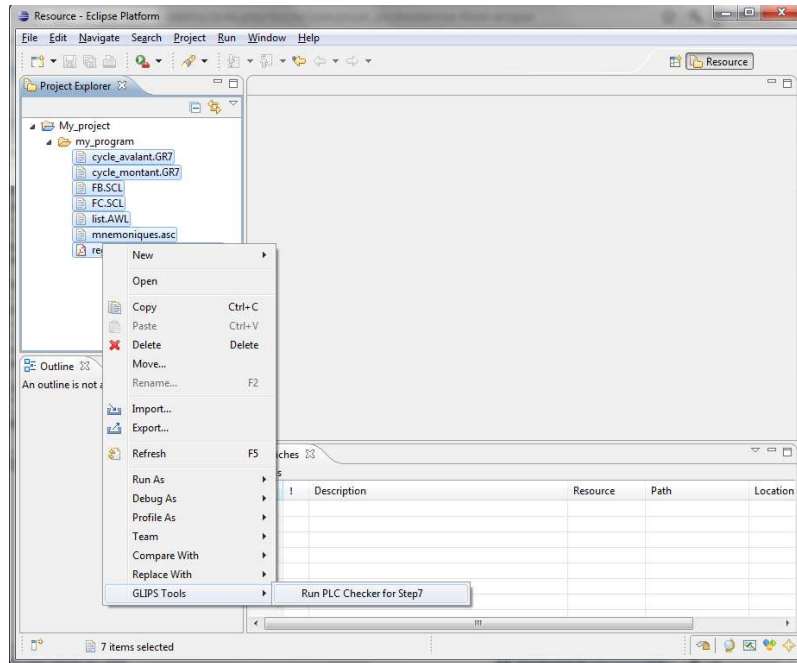


Figure 4.28: Launch PLC Checker via Eclipse platform

- If some files contains unsaved modifications, then the following dialog box appears.
 - Clic on **Deselect All**, if you DO NOT want to take account of the unsaved modifications for this analyze. Note that the unsaved modifications wont be deleted.
 - If you want to save modifications before to launch the analyze, then clic on **Select all**.
 - Clic on **OK** to continue. Clic to "Cancel" to cancel the analyze.

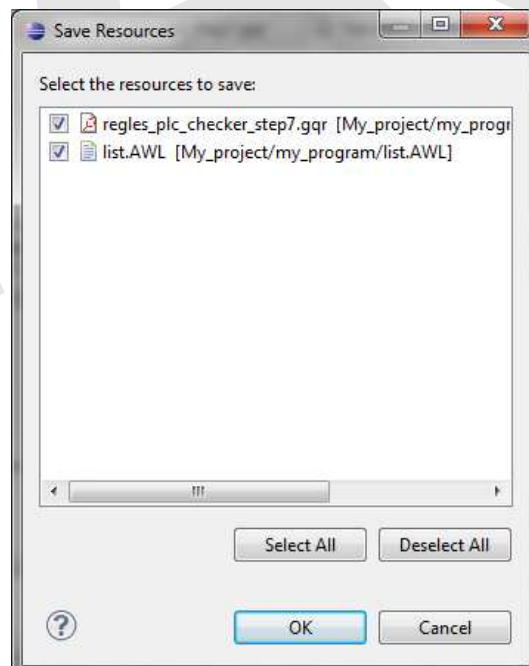


Figure 4.29: Save files modifications before the analyze

- A dialog box opens to save the result file (. gqr). We have the ability to rename it or change its directory.

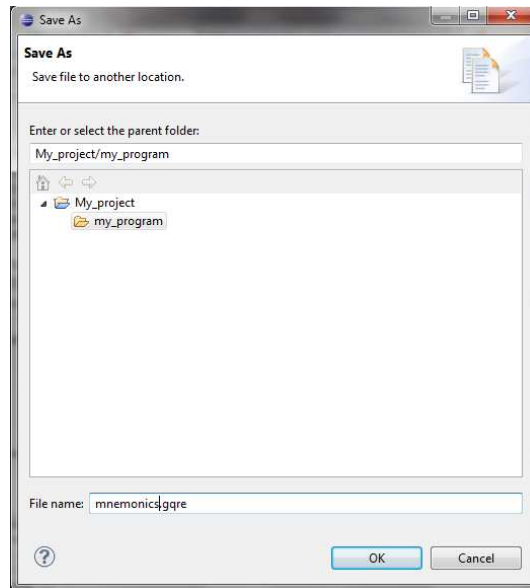


Figure 4.30: Save rules file modifications before the analyze

- During the analyze, a dialog box allow you see progression and to "Cancel" execution. The analyze phases are:
 - Sent files (application and rules) to the IAS server,
 - Read application file,
 - Analyze application file following rules file,
 - Update results,
 - create results file on Eclipse workspace,
 - Open results file with result viewer embedded in the Eclipse platform.

4.2.2 Run PLC checker from the website:

- Download the PLC application export file(s) on the server from the website (step 2).
1. If you are not at step 2, clic on the corresponding link **"Upload files with the wizard below"**.
 2. Search on your disk the mandatory and optional files using the **"File"** field
 3. Clic on the **"Add all files"** button to upload files on the IAS server.

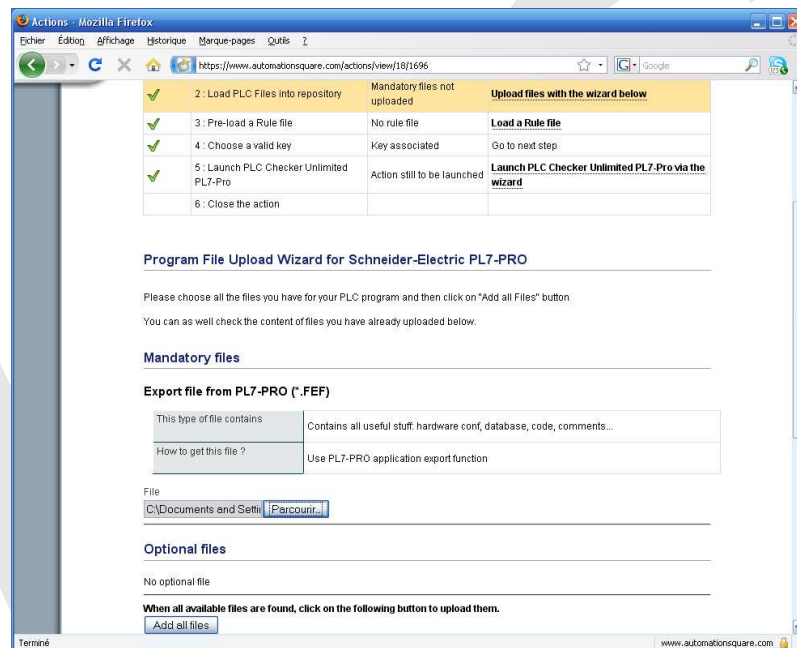


Figure 4.31: Select PLC application export file(s)

- Download the rules file on the server from the website (step 3).
1. If you are not at step 2, clic on the corresponding link **"Load a rules file"**.
 2. Search on your disk the rules file using the **"File"** field
 3. Clic on the **"Add all files"** button to upload files on the IAS server.

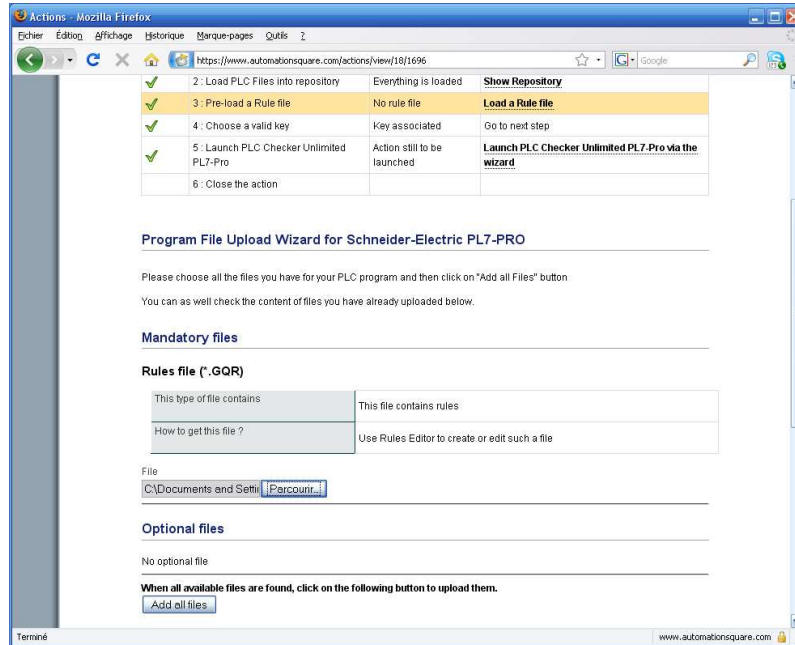


Figure 4.32: Select rules file

- Run PLC Checker on the server from the website (step 5).
 1. If you are not at step 5, clic on the corresponding link **"Launch PLC Checker ... via the wizard"**.
 2. Launch the analyze using the **"Launch PLC Checker"** button.

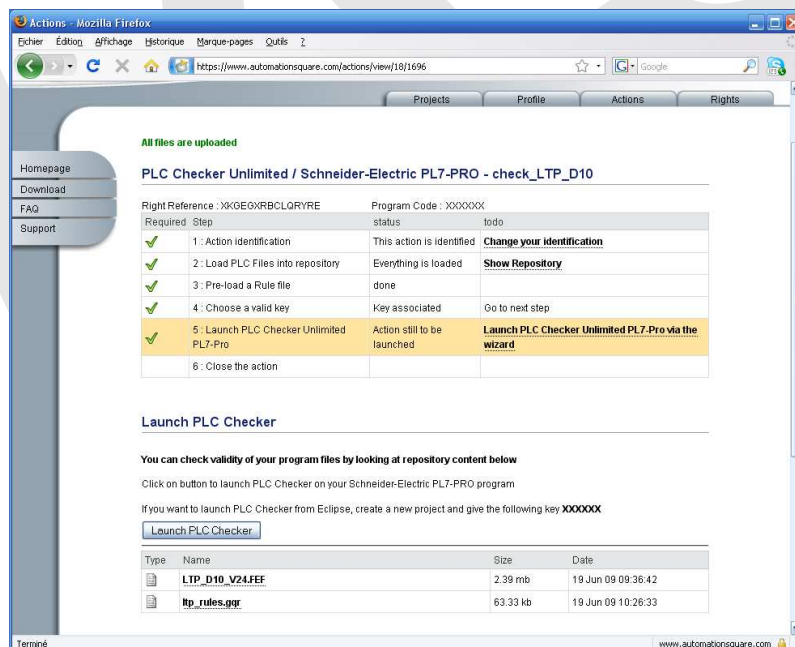


Figure 4.33: Launch PLC Checker via the wizard on the website

4.3 View PLC Checker results

4.3.1 View results on Eclipse platform

- Get last action results (**ONLY in the case of :existing PLC Checker project creation on Eclipse platforme**):
 1. **create a new project** on the Eclipse platform (see Existing PLC Checker project creation on Eclipse platforme) getting files and characteristics of the project located on IAS server and previously created via the web-site.
 2. **Select the rules file**
 3. open the **right-click** contextual menu
 4. Get the last action result using this contextual menu: **GLIPS tools / Get last action results**

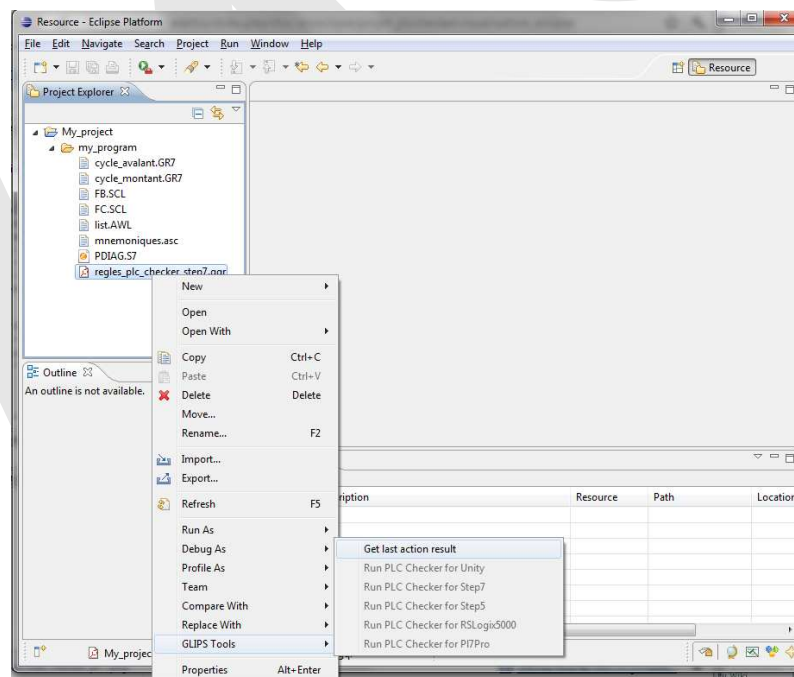


Figure 4.34: Get last action results

- View result file (*.gqre) on Eclipse platform.
 - When asked to, by the Eclipse platform, you have to choose the result file name. Clic on **OK**.

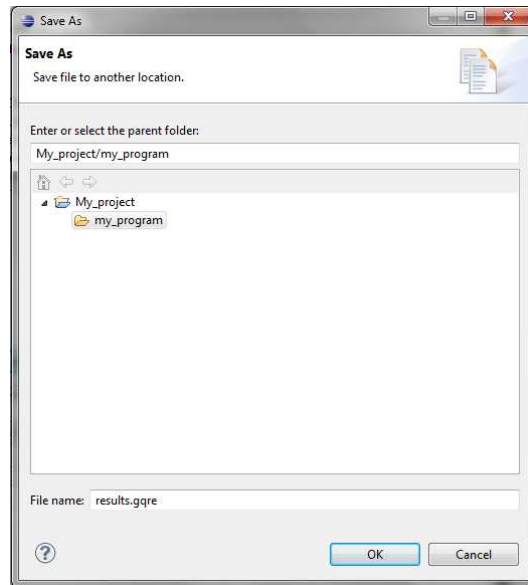


Figure 4.35: Result file name

- If the result file already exists the following dialog box appears.
 - Clic on **Yes** to delete the existing result file.
 - Clic on **No** to go back to the result file name definition.
 - Clic on **Cancel** to cancel getting results.
- The result file (*.gqre) is displayed on the result viewer embedded in the Eclipse platform.

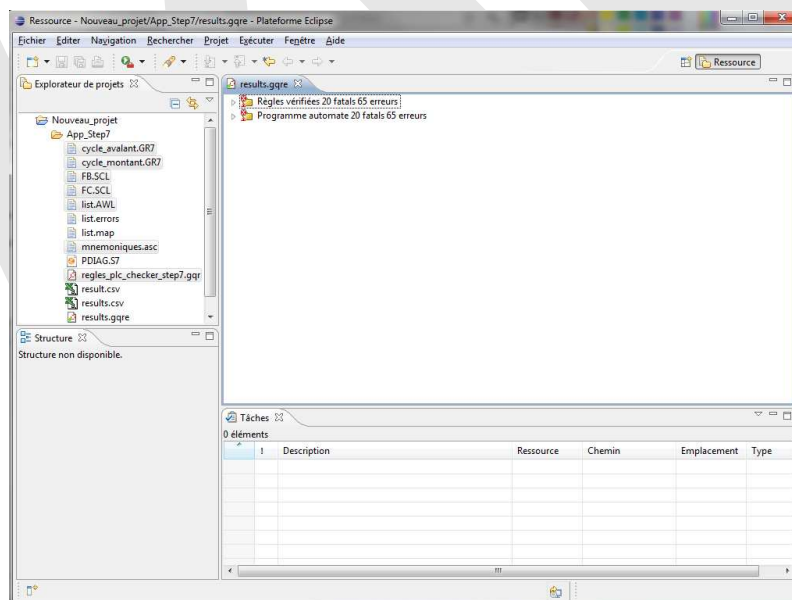


Figure 4.36: Result File

Chapter 5

PLC export files generation

5.1 FEF export file generation from PL7-Pro

PL7-Pro application must be exported in a FEF export file in order to be analyzed by PLC Checker.

You can create an FEF export file with the **File > Export application...** menu in PL7-Pro software.

1. In PL7-Pro menu, select **File > Export application...**
2. In the "Export application" dialog box, give a **name** to the FEF file
3. DO NOT change directory where the FEF file will be created, and clic on **Save**

Before to be analysed by the PLC Checker, the FEF file must be imported in the Eclipse platform workspace.

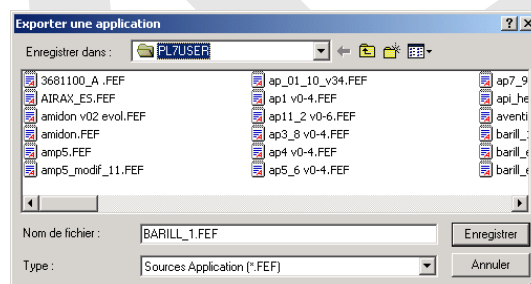


Figure 5.1: Export FEF file

5.2 L5K export file generation from RSLogix5000

RSLogix5000 application must be exported in a L5K export file in order to be analyzed by PLC Checker.

You can create an L5K export file with the **File > Save as...** menu in RSLogix5000 software.

1. In RSLogix5000 menu, select **File > Save as...**
2. In the "Save as" dialog box, give a **name** to the L5K file
3. choose the file type : **RSLogix 5000 Import/Export File (*.L5K)**
4. DO NOT change directory where the L5K file will be created, and clic on **Save**.

Before to be analyzed by the PLC Checker, the L5K file must be imported in the Eclipse platform workspace.

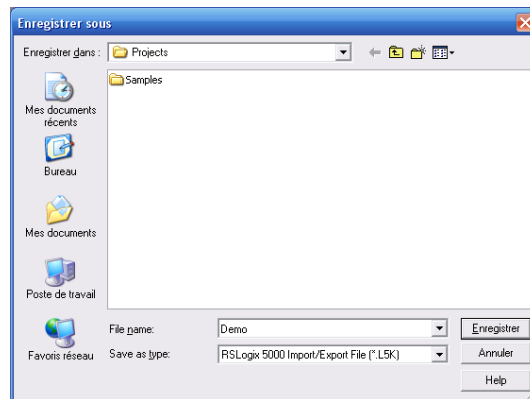


Figure 5.2: Export L5K file

5.3 ASC, AWL, GR7 and SCL Export files generation from Step7

To analyze a STEP7 application with PLC Checker, you have to:

- create and export the symbols of this application in an **.ASC** file, using the German abbreviations,
- create and export the contact code (CONT) in an **.AWL** file,
- create and export all graph blocs (GRAPH) in **.GR7** files,
- export structural codes (SCL) in **.SCL** files.

5.3.1 Create ASC file

- To create an **.ASC** file, you have to open the symbols editor.

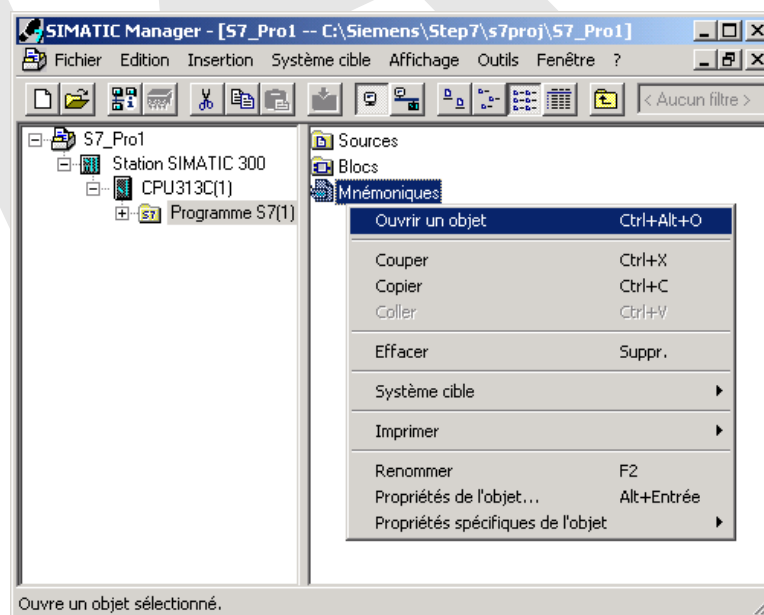


Figure 5.3: Open symbol table

- In the symbols editor, use the **Table > Export...** menu

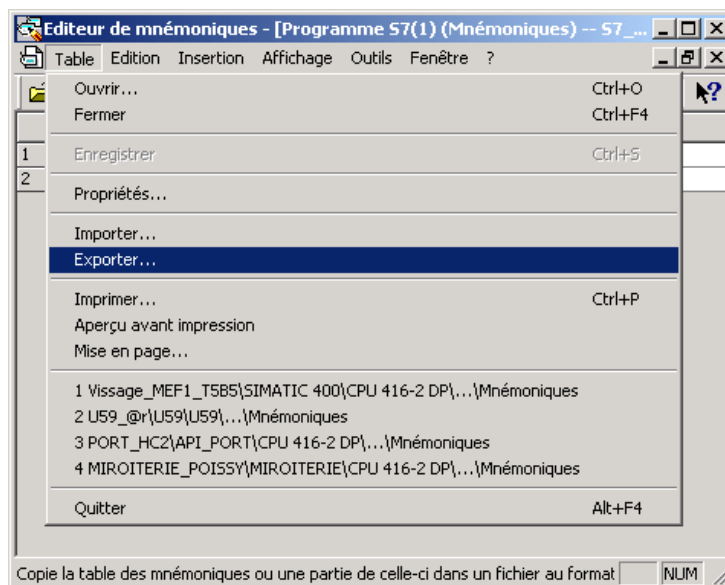


Figure 5.4: Export the symbol table

- Choose the **ASCII¹ Format (*.ASC)** file type, the **name** and the **directory** of the **.ASC** file to create, and clic on **Save**.

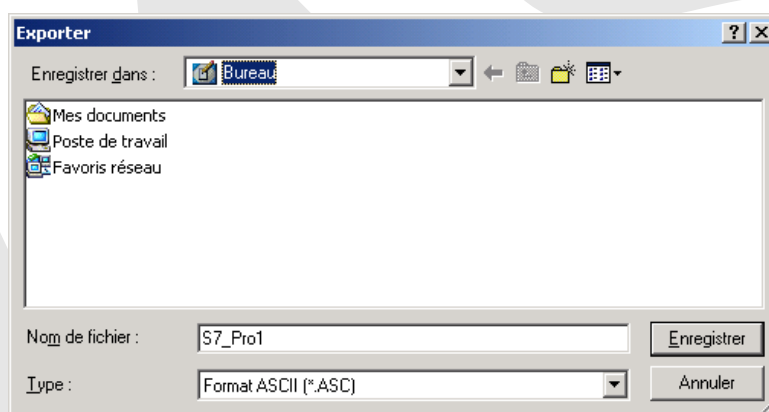


Figure 5.5: Define .ASC file name

¹American Standard Code for Information Interchange

5.3.2 Create AWL file

- To create the **.AWL** file, you have first to insure that your STEP7 workshop uses the German abbreviations.(menu **Tools > Parameters...**)

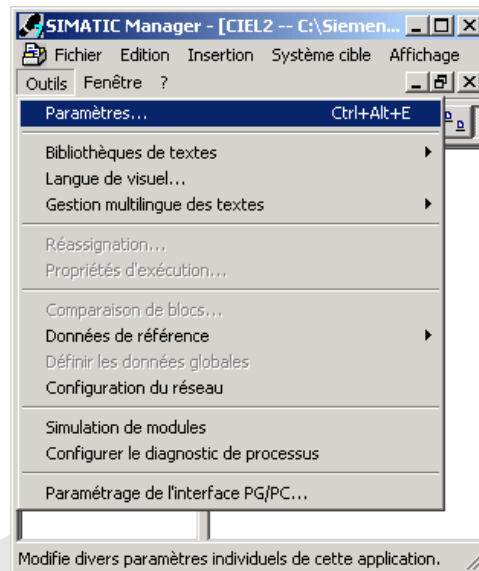


Figure 5.6: STEP7 workshop parameters

- In **Language** tab, select **German abbreviations** and clic on **OK**.

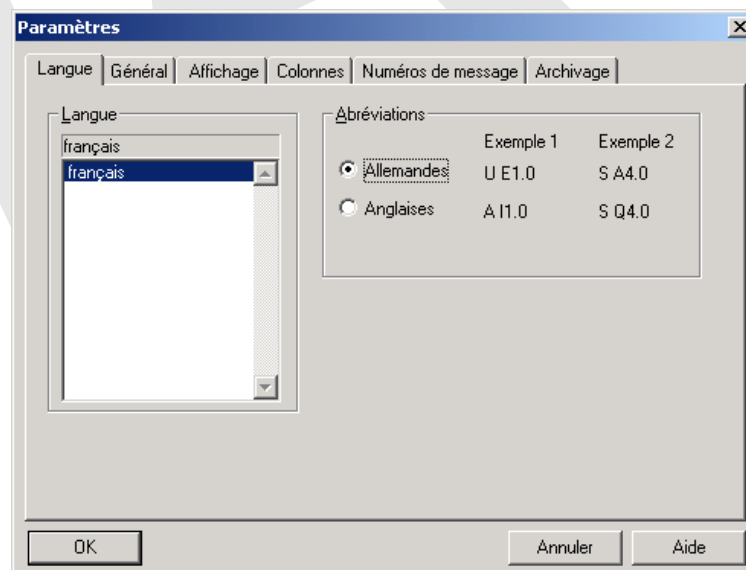


Figure 5.7: Select German abbreviations

- To create the **.AWL**file, source LIST must be generated. Open CONT/LIST/LOG code editor.

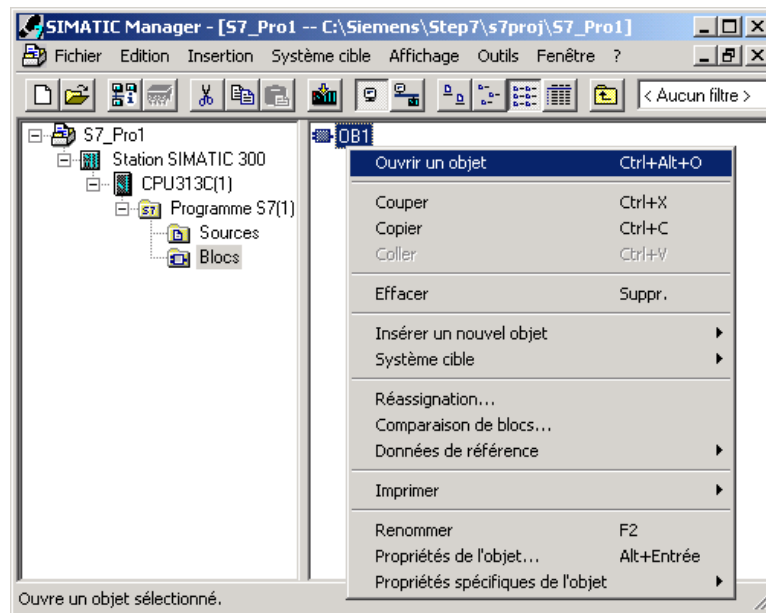


Figure 5.8: Open CONT/LIST/LOG editor

- Close all edited objects (Menu **File** > **Close**)

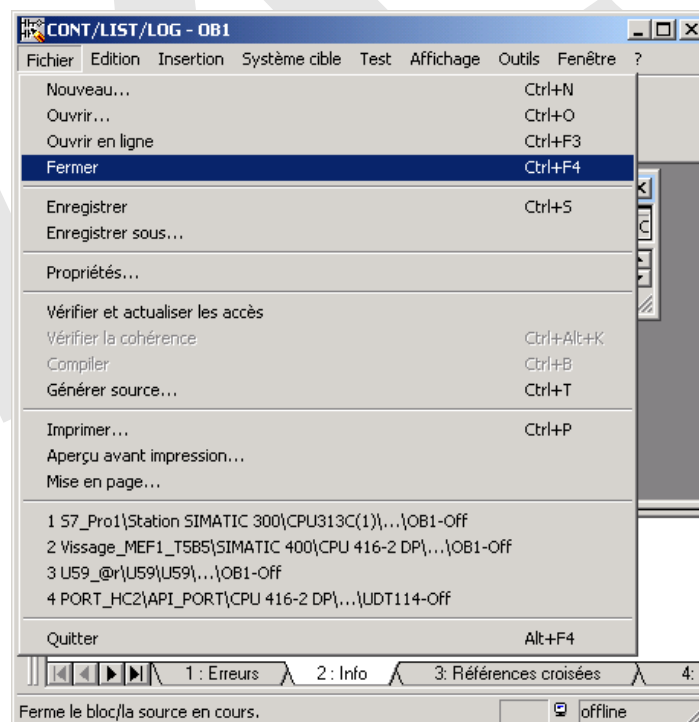


Figure 5.9: Close all edited objects

- Generate the program source (Menu **File** > **Generate source...**)

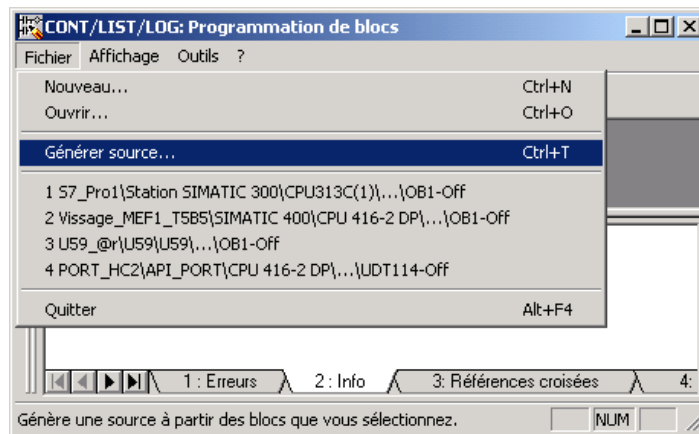


Figure 5.10: LIST source generation

- Give a name of the object to be generated.

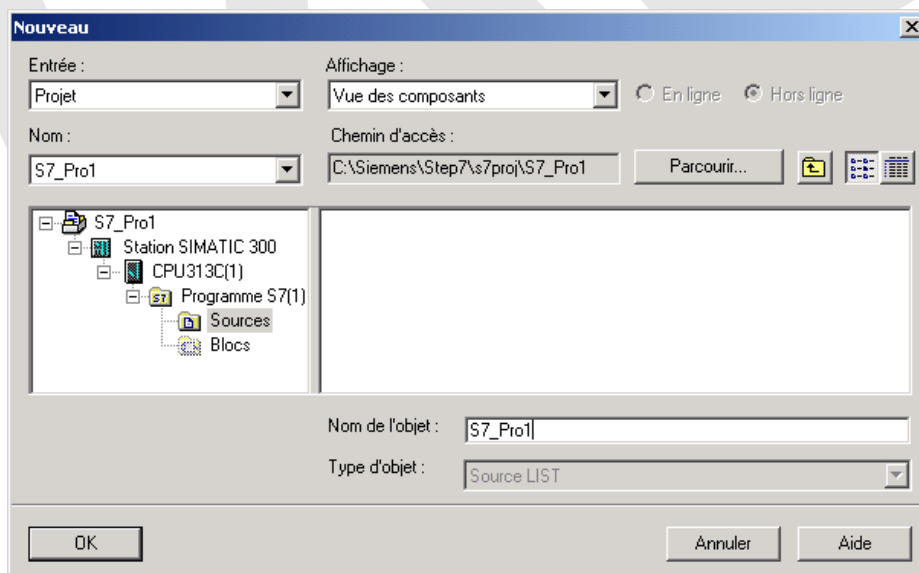


Figure 5.11: Name of .AWL file

- Use button **All > select all blocs**, select only the **Sort by program structure** option. Select the **symbolic operands** and clic on **OK**.

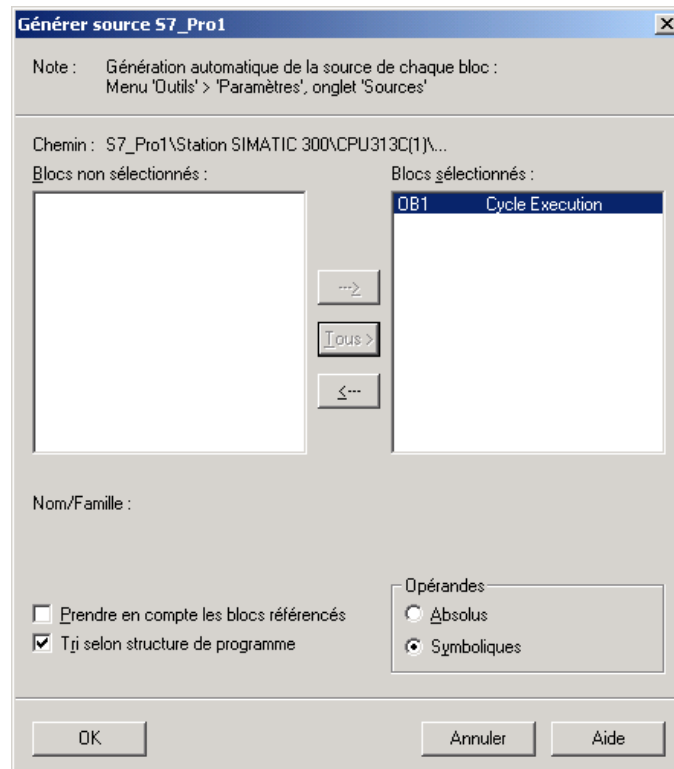


Figure 5.12: Configure source to be generated

5.3.3 Create GR7 file

- To create the GR7 sources, select all GRAPH blocs and open the GRAPH editor

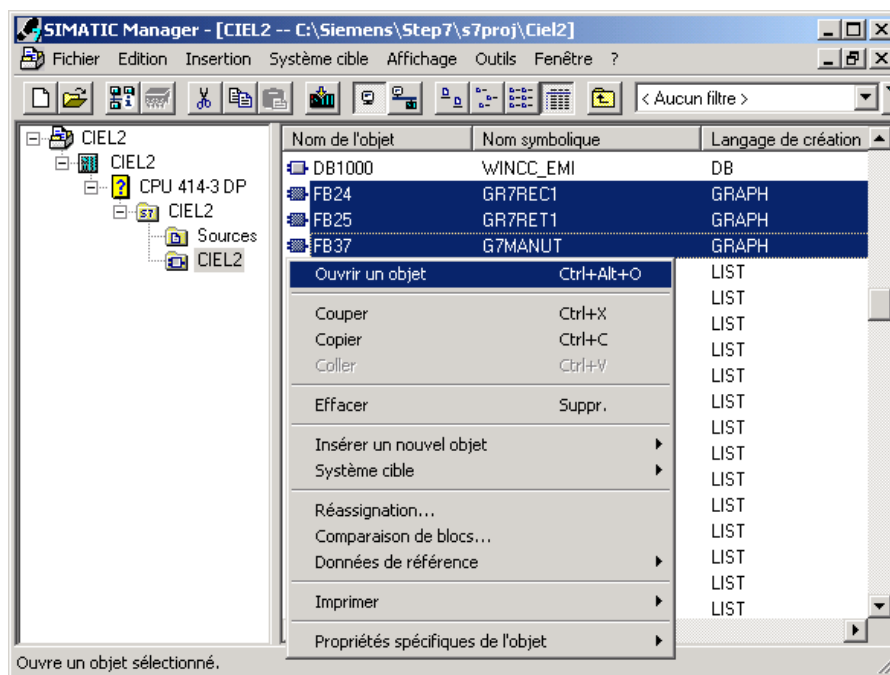


Figure 5.13: Open all GRAPH blocs in GRAPH editor

- For each GRAPH bloc, generate the source (menu **File** > **Generate a source**).

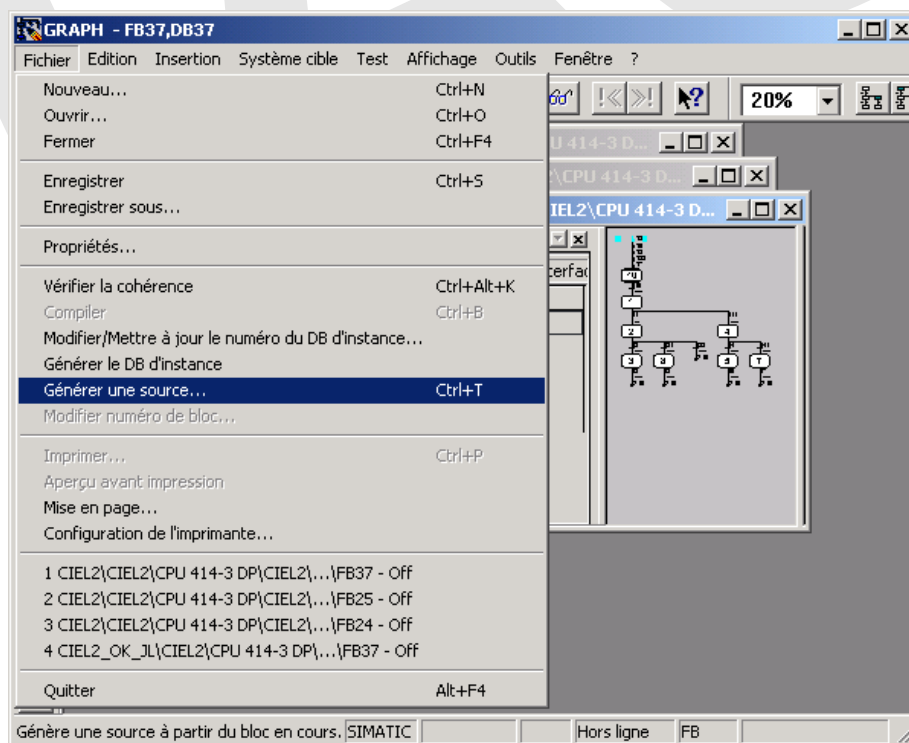


Figure 5.14: Generate each GRAPH source

- Give a name to each GR7 source file and press **OK**.

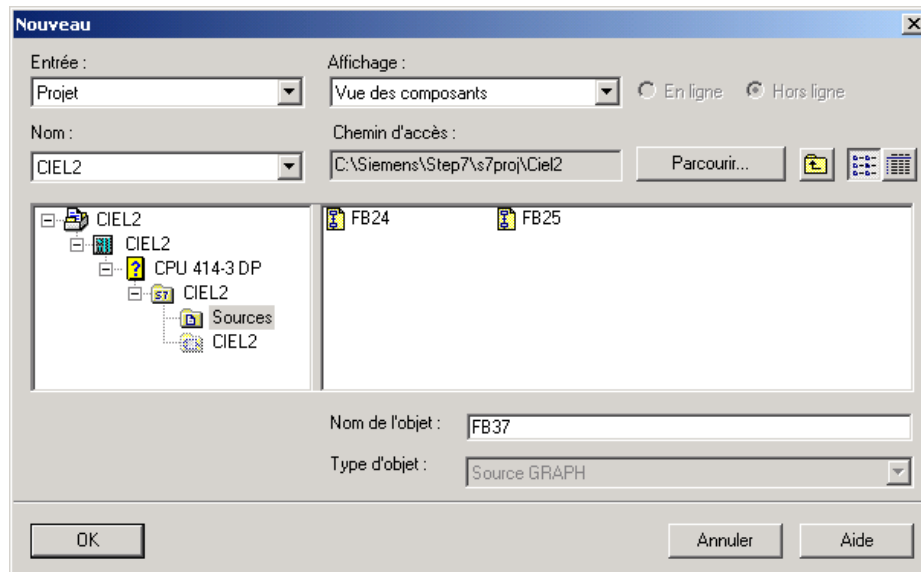


Figure 5.15: Name of GR7 files

5.3.4 Export files

- Select LIST and GRAPH sources (resp. AWL and GR7 files) previously generated and also the SCL sources. Copy sources using contextual menu **Export source...**

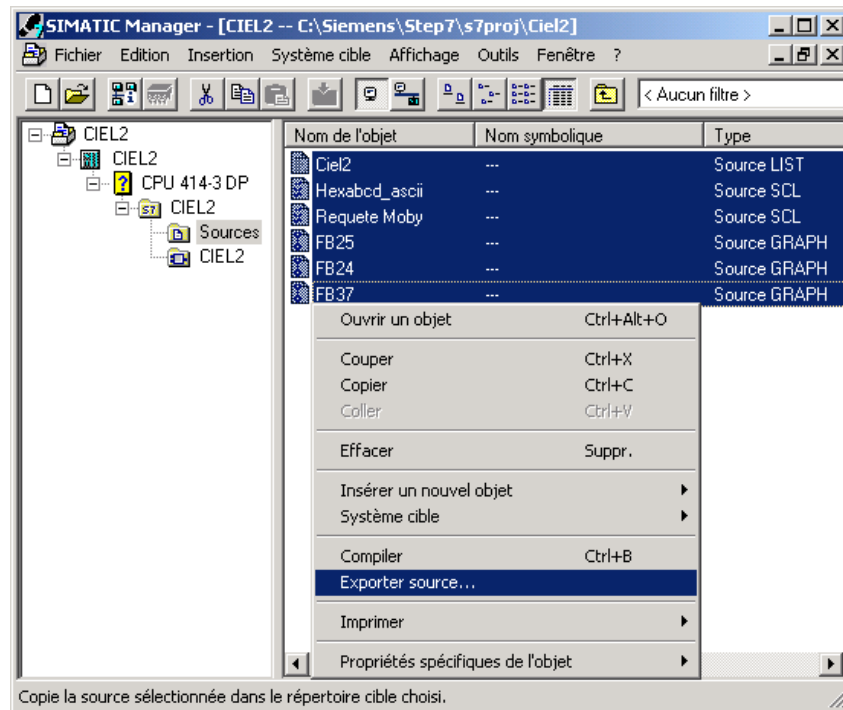


Figure 5.16: Select sources to export

- **Select folder** where to export sources files.

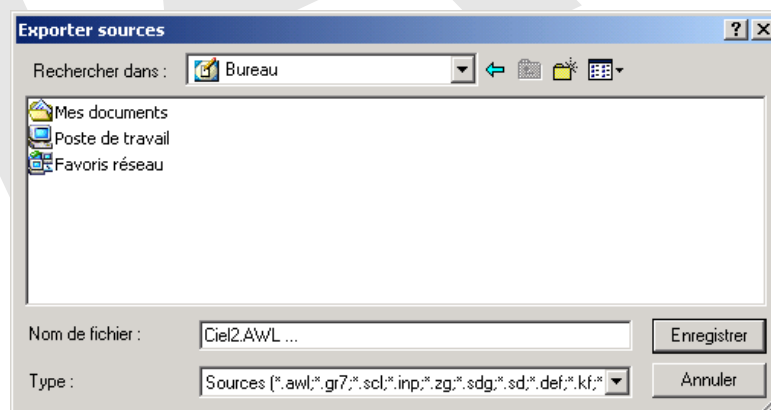


Figure 5.17: Select destination folder

Before to be analyzed by the PLC Checker, the **ASC**, **AWL**, **GR7** and **SCL** files must be imported in the Eclipse platform workspace.

5.4 XEF export file generation from Unity

Unity application must be exported in a XEF export file in order to be analyzed by PLC Checker.

You can create an XEF export file with the **File > Export application...** menu in Unity software.

1. In Unity menu, select **File > Export application...**
2. In the "Export application" dialog box, give a **name** to the XEF file
3. DO NOT change directory where the XEF file will be created, and clic on **Save**

Before to be analysed by the PLC Checker, the XEF file must be imported in the Eclipse platform workspace.

Note that, for unity v6.0 also, the export file must be a XEF file and not a ZEF file which is the default export format of Unity v6.0.

VAS

Chapter 6

Concepts

6.1 Rules

This section introduce part of the concepts used for the rules files.

6.1.1 Generality

A rules file has a hierarchic structure that can be represented by a tree. The nodes of this tree are rules groups (rulesGroup). The leaves of this tree describe the rules to check or are rules groups used to define reusable filters.

6.1.2 Rules activation

The nodes and leaves of rules file can be activated or deactivated using the associate box displayed at their left. A deactivated node (and his children) will not be read nor used during PLC Checker execution. By this way, with rules file split in independent parts, it is possible to check only some parts of this rules file with PLC Checker.

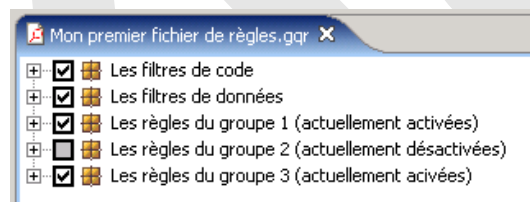


Figure 6.1: Rules activation

6.1.3 Rules name

This property is named "name".

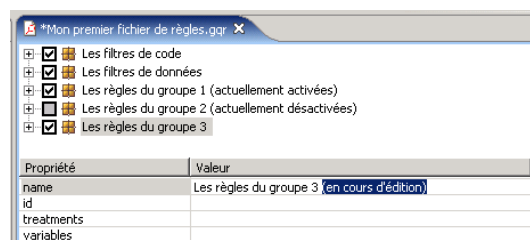


Figure 6.2: Rules naming

Each element of the rules file has a name. This is a comment that tells what the element represents. To change it, enter the name property.

Note: This name is used in the results file in the "Checked rules" section. You better have to systematically name each rules file element.

6.1.4 Rules Identity

This property, named "**id**", is used when you want to reuse, elsewhere in the rules file, the set of code or data defined by this element (via treatments and/or variables properties).

- For example, you can define the set of all remote control variables in your application.
 1. create a rules group in your rules file that will represent this set.
 2. Define his name (name property) "Define set of remote control variables".
 3. Define his ID (id property) "Remote_Ctrl".
 4. Define the set of remote controls via one or more address intervals or variable name formats in the "variables" property.
 5. When you need to add a rule that check all remote controls, you will use #Remote_Ctrl as variables property for your new rule.

Note: The ID property must be limited to alphanumeric characters (A..Z a..z 0..9) and underscore (.). The special characters are not allowed (+ - # @ : . ? * ...).

6.1.5 Code filter

This property, named "**treatments**", allows to define set of codes selected from the whole application code.

6.1.6 Data filter

This property, named "**variables**", allows to define sets of data from the whole application data.

6.1.7 To enable a rule using other rules

This property, named "**enabledIds**", contains a boolean expression of rules IDs. This allows to validate (statically, not dynamically) a rule or a rules group using one or more other nodes previously defined.

A rule or rules group is enabled when this property is true. The rules IDs used in the boolean expression are evaluated as true or false depending if their are enabled or not.

This boolean expression supports the OR, AND and NOT operators but does not support parentheses. The standard notation (ID1 OR ID2) and the post-fixed one (ID1 ID2 OR) are supported.

The rules ID used in the boolean expression have to be defined before their use in the enabledIds property.

- **Examples:** Here are some cases for the enableIds property of rule named rule3, with ID1, ID2 and ID4 the values of the ID properties of three previously defined rules named respectively rule1, rule2 and rule4:
 1. enabledIds="ID1 OR ID2" : rule3 is enabled when rule1 OR rule2 is enabled.
 2. enabledIds="ID1 ID2 OR" : rule3 is enabled when rule1 OR rule2 is enabled.
 3. enabledIds="ID1 AND ID2" : rule3 is enabled when rule1 AND rule2 are enabled.
 4. enabledIds="ID1 ID2 AND" : rule3 is enabled when rule1 AND rule2 are enabled.
 5. enabledIds="ID1 ID2 ID4 OR AND" : rule3 is enabled when rule1 AND (rule2 OR rule4) are enabled.
 6. ~~enabledIds="ID1 AND (ID2 OR ID4)"~~ : The parentheses are not supported. Unexpected result may occurs.
 7. enabledIds="" : rule3 may be enabled or not by the end user, using the check box(see rule activation section above).
 8. enabledIds="true" : rule3 is always enabled, the final user can not use the check box to disable this rule.
 9. enabledIds="false" : rule3 is always disabled, the final user can not use the check box to enable this rule (may be you better have to delete this rule, no ?).

6.1.8 To enable one rule or rules group in a list

This operation uses the **"groupIds"** property.

This **"groupIds"** property contains a rules ID list.

This allows to enable one and only one rule or rules group of a list.

- **Example:**

- The three nodes whose ID are ID1, ID2 and ID3, have the following groupIds property: groupIds="ID1 ID2 ID3". During the rules file edition, the rules editor plugin ensure that only one of the three nodes is enabled.

6.1.9 Identifier for .csv format export

This property, named **"csvId"**, allows to add 4 columns in the csv ("Comma Separated Value") file which is generated by the Eclipse platform when creating the result file.

Those columns contain:

- the counter of all occurrences founded under the corresponding node
- the errors counter
- the warnings counter
- the infos counter

6.1.10 Node visibility for protected files

This property, named **"visible"**, allows to mask, when false, the node and his sub-tree during rules file edition when this rules file is protected (locked property = "true" for the first node of the rules file).

6.1.11 List of editable properties for protected files

The property, named **"editable"**, contains the list of properties that can be changed when the rules file is protected (locked property = "true" for the first node of the rules file)

6.1.12 Errors severity

This property, named **"severity"**, allows to define the way the errors are added to the result file.

The value can be "fatal", "error", "warning", "info" or "nolog".

The value "nolog" allows to only count the objects (code or data) corresponding to one or more specification defined by one or more rules.

6.1.13 "warningThreshold" property

The **"warningThreshold"** property is reserved for future usage.

6.1.14 "errorThreshold" property

The **"errorThreshold"** property is reserved for future usage.

6.1.15 "defaultResultId" property

The **"defaultResultId"** property defines where to display the error in the PLC application tree of the result file.

6.1.16 "errorMessage" property

The **"errorMessage"** property defines the error message to be displayed in case of error. This property can contain variables between characters `"` and `"`. When error message is emitted, those variables are replaced by their values.

Here are some usefull variables (respect the uppercase):

- **FUNCTION**: contains the name of the bloc of code related to the error.
- **LOCATION**: contains the name and the offset of the bloc of code.
- **VARIABLE**: contains the name of the variable related to the error.
- **STR**: is an additional information depending on the checked rule.

6.1.17 "noMatchMessage" property

The **"noMatchMessage"** property, defines the message to be displayed when the set of data or code, on which the rule is checked, is empty.

6.2 Filters

This section introduce the filters concept used in the rules file.

6.2.1 Generality

The filters allow to define sub-set of elements.

Two elements set are taken into account: the set of the application code, and the set of the application data. So there is two kind of filters, the code filters and the data filters:

- The code filters are defined using the **treatments** properties of the rules file elements.
- The data filters are defined using the **variables** properties of the rules file elements.

For each element to be checked by PLC Checker, the analyze is limited:

- to the sub-set defined by his **treatments** property, for the codes
- to the sub-set defined by his **variables** property, for the data.
- Example: A rule, verifying a variable is never written, gives the set of variables, in the variables sub-set defined by the **variables** property, never written in the sub-set of code defined by the **treatments** property.

6.2.2 Limitations

The **treatments** and **variables** properties can not contain space character (" ") and must begin by the "+" or "-" operator.

6.2.3 Symbols filters

A symbols filter allows to define sub-set of objects using the symbol of those objects (also called mnemonic). The generic characters "*" and "?" are authorized and have the usual signification.

- **Example:**
 - with filter `DF_*`, the `DF_HIGHTEMP` and `DF_LOWTEMP` are selected in the sub-set
 - with filter `T?_*`, the `TC_ON` and `TR_TEMP_WATER` are selected in the sub-set

6.2.4 Symbols filters using regular expressions

This kind of symbols filter allows to define sub-set of objects using the symbol of those objects (also called mnemonic) and using more powerful syntaxes than simple symbols filters.

This kind of filter is defined by a regular expression into parentheses "()" . The regular expression describes the sub-set of symbols to be selected. The regular expression can contains meta-characters and quantifiers.

Meta-characters:

- . : all characters. **Ex:** (t.t.) $\Rightarrow$ toto, tati etc...
- — : alternative. **Ex:** (tata—titi) $\Rightarrow$ tata or titi
- () : group. **Ex:** (t(i—ou)ti) $\Rightarrow$ titi or touti
- [] : characters class. **Ex:** ([a-cD-F]g) $\Rightarrow$ ag, bg, cg, Dg, Eg ou Fg
- ^ : allow to define unauthorized characters. **Ex:** ([^a-y]z) $\Rightarrow$ az, bz, ..., yz are not OK.
- \ : allow to use a Meta-character or a quantifier as simple character. **Ex:** (\. \— \(\) \[\] \\)\Rightarrow . — () [] \

Quantifiers:

- * previous set of characters can appears **0** or **more** times. **Ex:** (.*) $\Rightarrow$ all string of characters even an empty string
- + previous set of characters can appears **1** ou **more** times. **Ex:** ([0.9]+) $\Rightarrow$ all numbers from 0 up to 9999999...
- ? previous set of characters can appears **0** ou **1** time. **Ex:** (test([a-z])?) $\Rightarrow$ test, test_a, test_b, ... test_z

6.2.5 Attributes filtres

Those attributes filters allow to define sub-set of objects using the an attribute of the object (Data or Code). The list of available attributes depends on the kind of object and on the PLC programming tool (UNITY-PRO, PL7-PRO, STEP 7, RSLOGIX).

An attribute filter always begins by the keyword "!" or "!".

The keyword "!" is used to search object that has the correct direct attribute value. The objects for which this attribute is not available are not selected.

The keyword "!" is used to search object that has the correct direct of inherited attribute value. If the attribute is not available for the object, then the attribute is also searched in the parents of the object.

The name of the attribute is put after the keyword.

The keyword "=" is used between the name of the attribute and the filter that defines the set of authorized values for this attribute.

The filter definition is into parentheses "()". If a regular expression is used in this filter, then the regular expression is also into parentheses "()" like for symbols filters.

Example: ^language=(+ladder) ou !comment=(+(*spare.*))

6.2.6 Absolutes addresses filters

An absolutes addresses filter defines a sub-set using the object address.

An addresses filter appears like this: @<memory plan name>:<addresses interval>

It always begins by the "@" character followed by the memory plan name where to find the object (ex: MW or KD). The ":" character is between the memory plan name and the addresses interval. The filter ended by the limits of the addresses interval.

There are four kind of possible syntax for the addresses interval:

- unique address (for example: @MW:10)
- finite interval (for example: @MW:0..100)
- interval without upper limit (for example: @MW:100..*)
- the whole memory plan (for example: @MW:*)

6.2.7 Filters intersection

The data and code sub-sets used for the analyze correspond to the intersection of the sub-sets defined for this rule and of those of her parents.

- **Example:**
 - the A sub-set is the father of
 - * the B sub-set, which is the father of
 - the rule that works on the intersection of A and B sub-sets

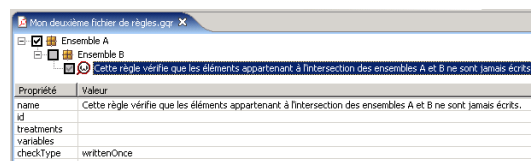


Figure 6.3: Filter intersection

6.2.8 Filters union

A sub-set can be defined as union of two or more sub-sets using the "+" operator.

- **Example:** +DF_+AL_+ER_*

6.2.9 Filters difference

A sub-set can be the difference of two sub-sets using the "-" operator.

In that case the +**A-B** sub-set is made of the **A** set from which the **B** set elements are removed.

- **Example:** +DF_*-DF_

6.2.10 Filters usage

To define a sub-set, rules file element of rulesGroup type are used.

This rulesGroup element must have an identifier (Id property).

To reuse this sub-set, use the "#" character before the element ID.

- **Example 1:**
 - rulesGroup named "digital inputs" as the ID: **dig.in**
 - rulesGroup named "analog inputs" as the ID: **ana.in**
 - a rule working on inputs can use the following variables property : **+#dig.in+#ana.in**
- **Example 2:**
 - rulesGroup named "digital inputs" as the ID: **dig.in**
 - rulesGroup named "analog inputs" as the ID: **ana.in**
 - a rulesGroup named "inputs", with ID=**all.in**, can regroup all inputs using the variable property **+#dig.in+#ana.in**
 - a rule working on inputs can use the following variables property : **+#all.in**

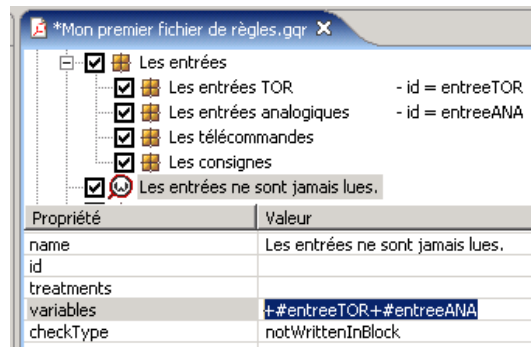


Figure 6.4: Re-use filter example

6.3 Result file

The results of the PLC Checker analyze, maid on the application file(s) using a rules file, are putted in a file with the **.gqre** extension (glips quality result).

In this file, the errors are ordered by two ways:

1. by rules order(using the checked rules tree of the rules file). Only the enabled rules are displayed.
2. and by application workshop order. Each error is attached to an element of the application workshop. In most of the cases, this element must be edited in order the error to be corrected.

6.3.1 Printing results file

To print or included the results file in a document, the results file must be open and then saved with **html** name extension.

The menu **File > Save as ...** displays the "save as" dialog box.

Change name extension in the File name field, replacing **gqre** by **html** (respect lowercase) and then clic **OK**. The **html** file is now in the Navigator view.

Then open the **html** file with your system editor (contextual menu **Open with > System editor**) and use his print (menu **File > Print...**) or copy (menu **Edit > Copy**) functions.

Chapter 7

The rules

1. Rules group (rulesGroup)
2. Read Check (readCheck)
3. Write Check (writeCheck)
4. Symbol check (symbolCheck)
5. Attribute check(attributeCheck)
6. Code blocks check (blockCheck)
7. Equation check (equationCheck)

7.1 Rules group (rulesGroup)



Figure 7.1: Rulesgroup

7.1.1 Usage

A rulesGroup element can be used:

1. to structure a rules file. It is only used to regroup the rules. For example a rulesGroup element can regroup all code filters, another one can regroup all rules related to written data.
2. as part of a filter created by intersection of its parents and its children.
3. to define a data or code filter that can be reused using its identifier (ID property).

7.1.2 Properties

Name (name)

See Concepts - Rules - Rule name

Identifier (id)

See Concepts - Rules - Rule identifier

Code filter (treatments)

See Concepts - Rules - Code filter

Data filter (variables)

See Concepts - Rules - Data filter

7.2 Read Check (readCheck)



Figure 7.2: Readcheck

7.2.1 Usage

A readCheck element allows to check that the number of readings made in the codes sub-set (defined by treatments properties) on the variables sub-set (defined by variables properties) is not to big (maxRead property) nor to small (minRead property).

7.2.2 Properties

Name (name)

See Concepts - Rules - Rule name

Identifier (id)

See Concepts - Rules - Rule identifier

Code filter (treatments)

See Concepts - Rules - Code filter

Data filter (variables)

See Concepts - Rules - Data filter

Minimal number of readings (minRead)

A readCheck rule is often used with a minRead of 1. In that case, the variables, from the data filter, that are NOT read in the code filter, are added to the result file. This can be used to check the application modularity.

Note : If this property is empty, then no error is generated.

Maximal number of readings (maxRead)

A readCheck rule is often used with a maxRead of 0. In that case, the variables, from the data filter, that are read in the code filter, are added to the result file. This can be used to check the application modularity.

Note : If this property is empty, then no error is generated.

7.3 Write Check (writeCheck)



Figure 7.3: Writecheck

7.3.1 Usage

A writeCheck element allows to check that the writings, made in code sub-set (code filter) on variables sub-set (data filter), are done in the mode defined by the checkType property.

7.3.2 Properties

Name (name)

See Concepts - Rules - Rule name

Identifier (id)

See Concepts - Rules - Rule identifier

Code filter (treatments)

See Concepts - Rules - Code filter

Data filter (variables)

See Concepts - Rules - Data filter

Type of checking (checkType)

Written from one task only (writtenFromOneTaskOnly) This rule check that all writings of each variable of the variable sub-set, found in the code sub-set, can be executed from only one task. Example : a PL7-Pro application may contain the MAST task, a FAST task and one task per events. This rule is useful to identify critical variables due to concurrent accesses.

Written only once (writtenOnce) This rule checks that each variable of the variables sub-set (variables properties) is written only once in the code filter (treatments properties). This is useful for example to check that the outputs are written from only one equation.

Set only once (SetOnce) This rule checks that each variable of the variables sub-set (variables properties) is set only once in the code filter (treatments properties).

Reset only once (ResetOnce) This rule checks that each variable of the variables sub-set (variables properties) is reset only once in the code filter (treatments properties).

Written from one code block only (writtenFromOneBlockOnly) This rule check that all writings of each variable of the variable sub-set (variables properties), found in the code sub-set (treatments properties), are in the same block of codes. Example : In a PL7-Pro application the blocks of codes are the sections, the SR and the events. This rule is useful to check application modularity.

Written before read (writtenBeforeRead) This rule check that each variable of the variable sub-set (variables properties) is written before read in the code sub-set (treatments properties). This rule can check application reactivity.

Written in block (writtenInBlock) This rule check that each variable of the variable sub-set (variables properties) is written at least once in the code sub-set (treatments properties). This rule allows to check that all application variables are used.

Not written in block (notWrittenInBlock) This rule check that each variable of the variable sub-set (variables properties) is NOT written in the code sub-set (treatments properties). This rule is useful to check application modularity.

7.4 Symbol check (symbolCheck)

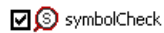


Figure 7.4: Symbolcheck

7.4.1 Usage

This rule allows to check:

1. that all application elements (data and code) have a symbol,
2. that the length of this symbol is not too short nor too long,
3. that the symbol does not contains reference to the physical address,
4. and that a group of elements (identify by the symbol) are in the given memory plan.

7.4.2 Properties

Name (name)

See Concepts - Rules - Rule name

Identifier (id)

See Concepts - Rules - Rule identifier

Code filter (treatments)

See Concepts - Rules - Code filter

Data filter (variables)

See Concepts - Rules - Data filter

Symbols minimal length (minLength)

This property defines for all application elements (code and data filters) the minimal length of its associated symbol.

Symbols maximal length (maxLength)

This property defines for all application elements (code and data filters) the maximal length of its associated symbol.

Symbol does not refer to absolute memory address (notAbsoluteCheck)

This boolean allows to select the application elements (code and data filters) whose symbol refers either to their absolute address or to their type.

- Examples :
 - W_10_MOTOR_SPEED at address %MW10 generates an error (reference to 10).
 - MW_MOTOR_SPEED at address %MW10 generates an error (reference to MW).
 - AMBIANCE at address %MB10 generates an error (reference to MB¹).
 - AMBIANCE at address %M10 does not generate an error (no reference test for type represented by only one character)
 - PUMP_101_PRESSURE at address %M10 does not generate an error (10 is compared to 101)

¹Megabyte

Variables belong to memory zone (memFilter)

This rule allows to check that each variable of the variables sub-set (variables properties) belongs to the sub-set defined by the memFilter property.

This rule is used, for example, to check that all remote controls are in the defined memory range.

7.5 Attribute check (attributeCheck)



Figure 7.5: Attributecheck

7.5.1 Usage

This rule allows to check the specified attribute (attribute property) only on selected elements types (see elements types selection below). The list of available attributes depends on element type and on PLC programming tool (Unity-Pro, PL7-Pro, Step7, RSLogix5000).

7.5.2 Properties

Name (name)

See Concepts - Rules - Rule name

Identifier (id)

See Concepts - Rules - Rule identifier

Code filter (treatments)

See Concepts - Rules - Code filter

Data filter (variables)

See Concepts - Rules - Data filter

Minimal length (minLength)

This property defines the minimal length of the value of the selected attribute.

Maximal length (maxLength)

This property defines the maximal length of the value of the selected attribute.

Attribute name (attribute)

This property defines the name of the attribute whose value has to be checked.

Attribute value expression (expression)

This property defines the expression that the attribute value has to match for no error to be generated.

Check only selected elements types

The following properties are used to select, when equal to true, the elements types whose selected attribute has to be checked by this rule:

- instruction
- typeDeclaration
- typeDesignation
- variableDeclaration
- variableDesignation
- prototypeDeclaration
- prototypeDesignation
- grafcetTransitionIncluded
- program
- grafcetStepIncluded
- label
- procedureDeclaration
- procedureDesignation
- instructionList
- operator
- rung

7.6 Code blocks check (blockCheck)



Figure 7.6: Blockcheck

7.6.1 Usage

This set of rules work on the code blocks. The checking depends on the checkType property (see below).
For example, code blocks for PL7-Pro applications are the sections and SR.

7.6.2 Properties

Name (name)

See Concepts - Rules - Rule name

Identifier (id)

See Concepts - Rules - Rule identifier

Code filter (treatments)

See Concepts - Rules - Code filter

Data filter (variables)

See Concepts - Rules - Data filter

Check type (checkType)

The block contains dead code (containsCode) This check type insure that each of the code blocks defined by the treatments property contains at least one instruction.

The block is present (present) This check type insure that at least one of the code blocks defined by the treatments property is present in the application.

This is used to check that application follows a given conception plan.

The block is NOT present (notPresent) This check type insure that each of the code blocks defined by the treatments property is NOT present in the application.

This is used to check, for example, that temporary codes (adjustment, simulation, validation) are removed from application.

The block contains dead code (containsDeadCode) This check type insure that each instruction of each of the code blocks defined by the treatments property can be executed in the application.

The set of the unreachable instructions is added to the result file.

This allows to find unused code that can be source of lost of time during maintenance.

7.7 Equation check (equationCheck)



Figure 7.7: Equationcheck

7.7.1 Usage

This rule works on writing references of variables. It allows to check whether at least one read reference of the X variable (var filter) is used or not in the instruction that writes the Y variable (variables filter).

For example, it is possible to check that all outputs (S_*) writings depend on an authorization variable (aut_*). ie: s_* = f(aut_*,...)

7.7.2 Properties

Name (name)

See Concepts - Rules - Rule name

Identifier (id)

See Concepts - Rules - Rule identifier

Code filter (treatments)

See Concepts - Rules - Code filter

Data filter (variables)

See Concepts - Rules - Data filter

This is the selection of the variables whose writing references will be checked.

Used variable filter (var)

This filter allows to define the set of variables whose read references are searched in the writing references of the variables defined by the previous filter (variables filter).

IAS

Chapter 8

Rules file creation and edition

8.1 Rules file creation

To create a rules files, follow this steps:

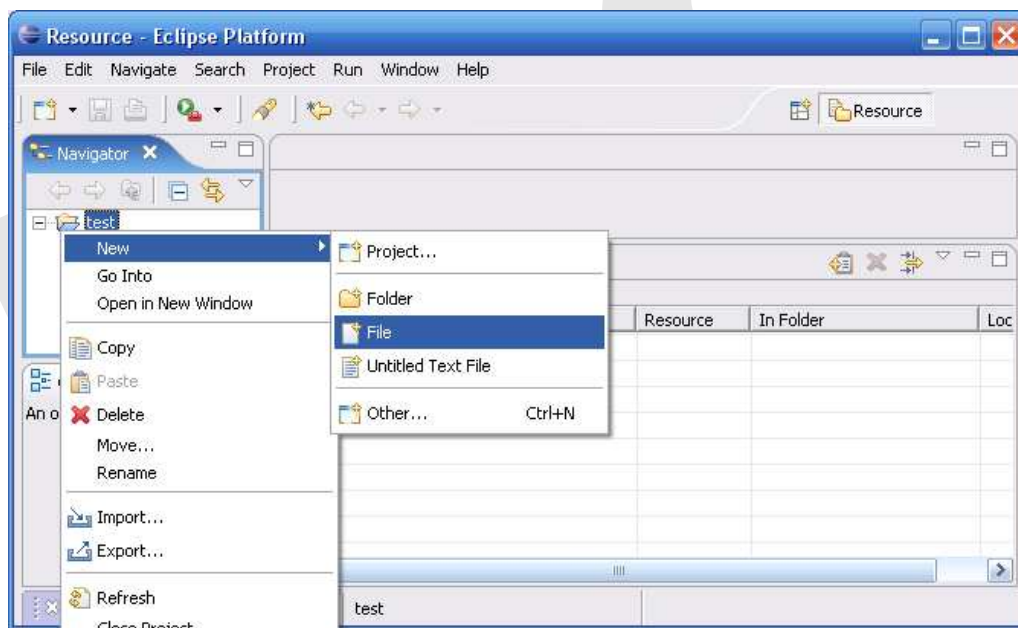


Figure 8.1: New file creation

1. In the Eclipse Navigator view, use **right clic** on the project or folder were you want to create the rules files.
2. In the contextual menu, select **New > File**.
3. Enter the file name with extension **".gqr"** (for example, My_rules_file.gqr). The extension must be **gqr** (glips quality rules) and must respect the lowercase.
4. Clic on **Finish**.

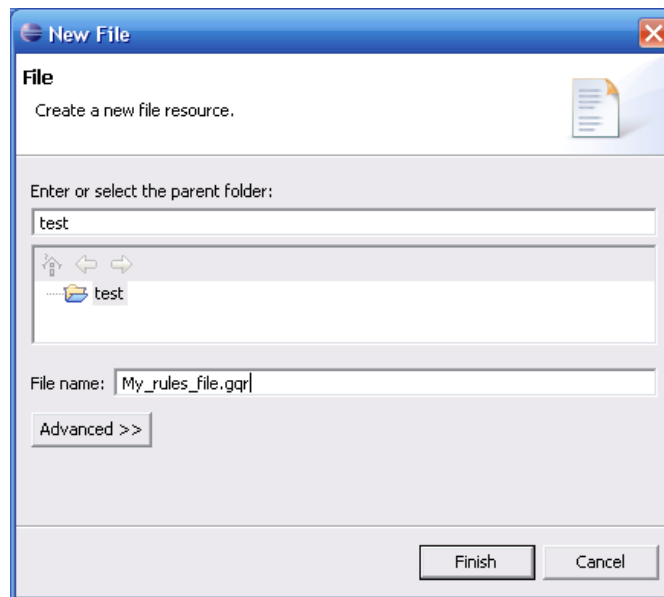


Figure 8.2: New rules file creation

A complete rules file is created. This file already contains basics rules.

The file is opened in the PLC Checker rules editor. You have to define the code sections (initialization, inputs acquisition, outputs emission, etc...), and also the variables types (inputs, outputs, alarms, etc...). If necessary, you'll have to define the addresses of the supervision tables.

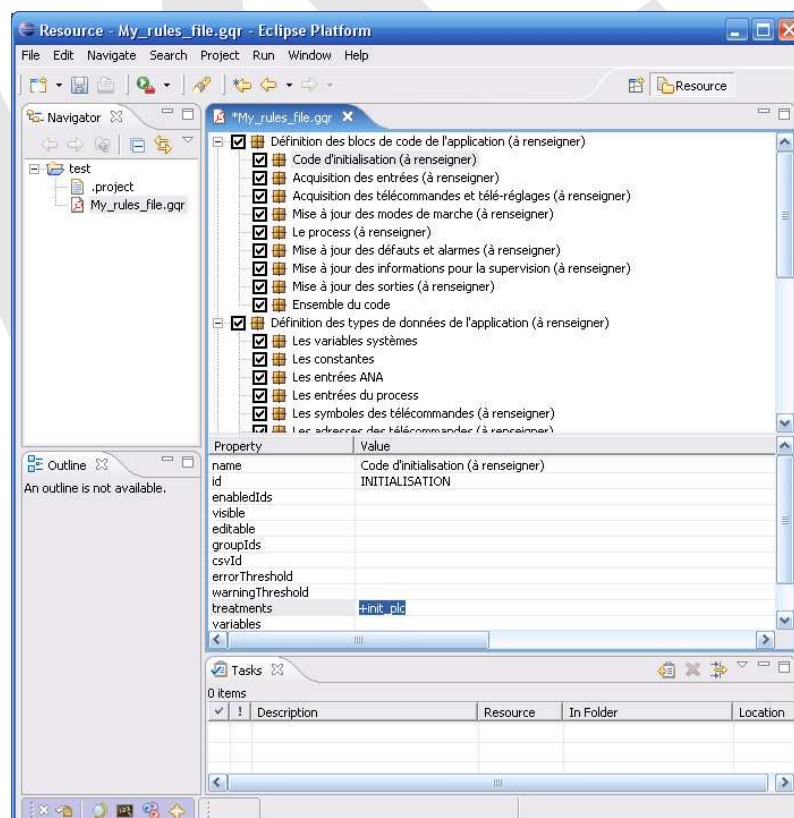


Figure 8.3: The new rules file

IMPORTANT Before to modify a treatments property (code sections) or a variables property (variables types), you'll have to read the Concepts - Filters section, in order to take account of the formalism used for those properties.

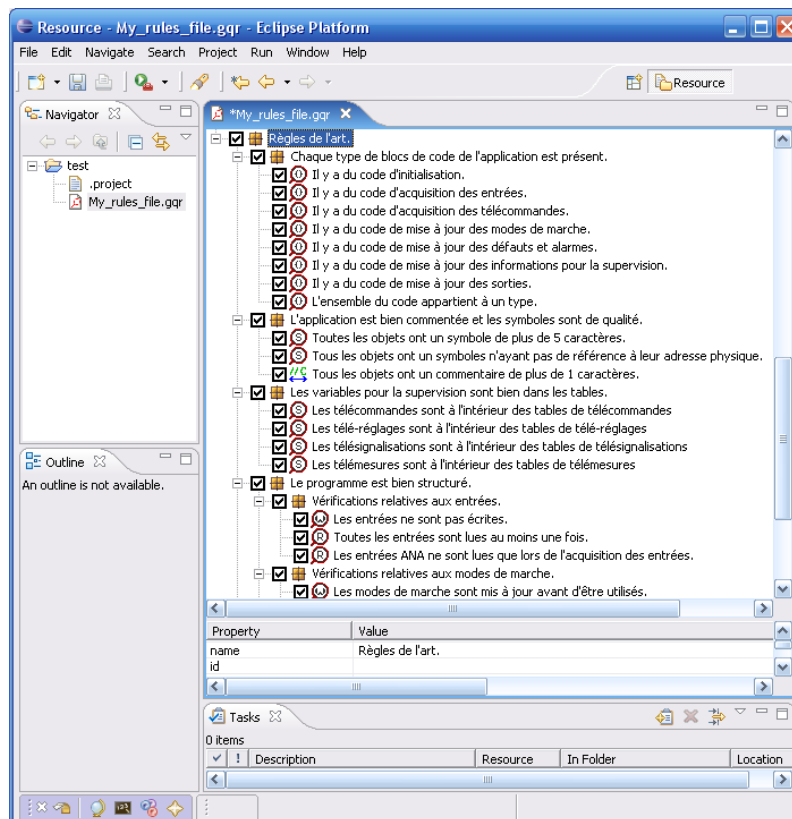


Figure 8.4: Basic rules included in the new rules files

8.2 Add a rule

All operations on rules are available in the contextual menu obtained by a **right clic** in the rules editor.

Use **New** to add a new rule.

- **New > Sibling > Before:** add a rule before the selected one.
- **New > Sibling > After:** add a rule after the selected one.

If the selected rule is a rulesGroup, then you can also add a child rule.

- **New > Child > First:** add a Child in head of the list of children of the selected rulesGroup.
- **New > Child > Last:** add a Child in tail of the list of children of the selected rulesGroup.

Then choose the rule type to be added (rulesGroup, writeCheck, symbolCheck, commentCheck, blockCheck, readCheck, ...).

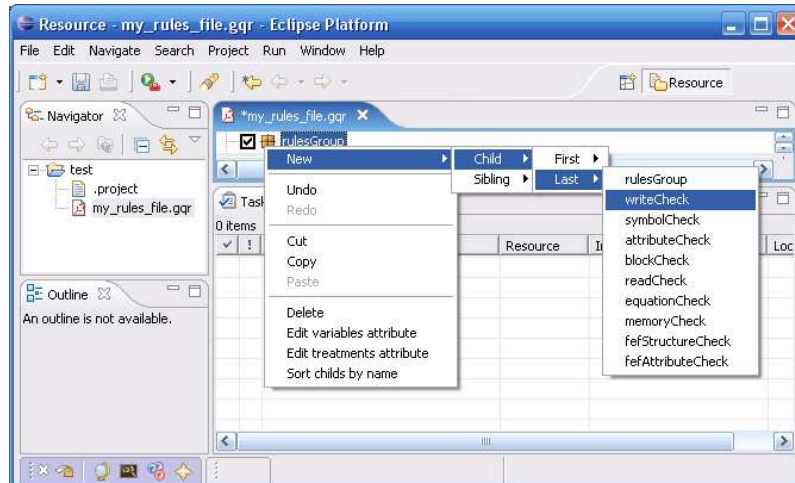


Figure 8.5: New rule creation

8.3 Cancel modifications

Use **Undo** in contextual menu, to go back and cancel modifications. You can redo modifications using the **Redo** in the contextual menu.

Note : When you come back up to the state saved on disk, the contextual menu **Undo** is disabled.

8.4 Copy and past rules

A set of contiguous rules with the same father can be copy using the **copy** command in the contextual menu.

The **Cut** command enable to copy and cut the selection. you can **Past** one or more times the previously copied selection. The rules selection is pasted after the rule selected.

Note : If the selected rule is a rulesGroup, and if it does not contain any child, then the rules selection is pasted as child of the selected ruleGroup.

8.5 Delete rules

use the **Delete** command in the contextual menu, in order to remove the rules selection. The rules selection is not copied.

8.6 Operations on rules properties

You can **copy**, **cut**, **past** properties contains, using the contextual menu displayed by a right clic on property field when it is edited (cursor displayed).

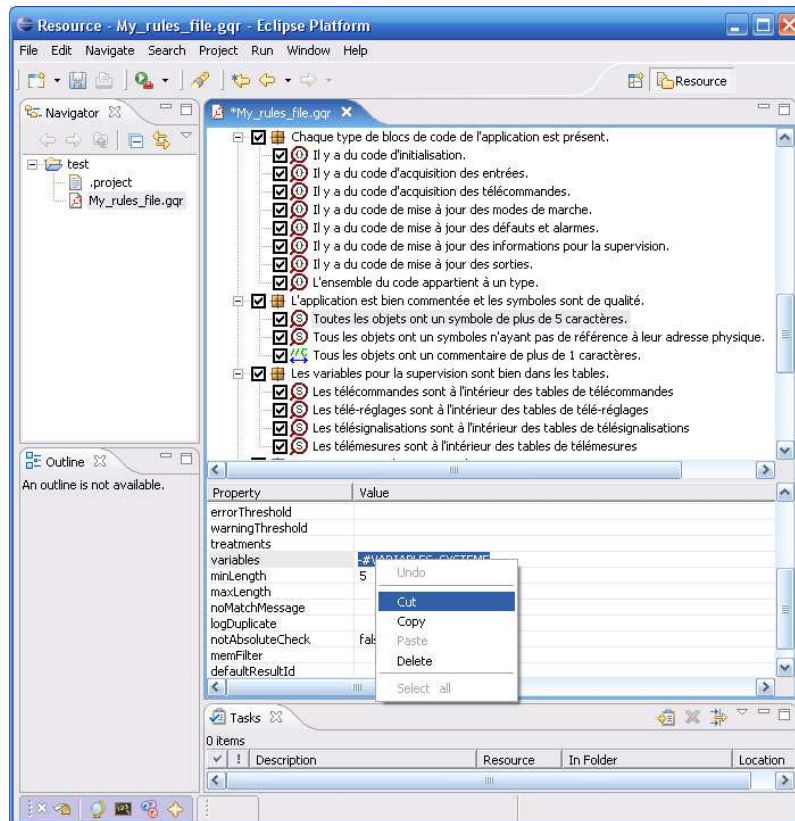


Figure 8.6: Copy-Past rules properties

8.7 Printing rules file

To print or included the rules file in a document, the rules file must be saved with name extension **html**.

The menu **File > Save as ...** displays the "save as" dialog box.

Change name extension in the File name field, replacing **gqr** by **html** (respect lowercase) and then clic **OK**. The **html** file is now in the Navigator view.

Then open the **html** file with your system editor (contextual menu **Open with > System editor**) and use his print (menu **File > Print...**) or copy (menu **Edit > Copy**) functions.

Chapter 9

Final license contract for PLC Checker

The license contract can be found at this address: <http://www.automationsquare.com/licenses.html>

IAS